OPEN SOURCE CNC CONTROL WITH CAM AND DIGITAL TWIN INTEGRATION

A Thesis
Presented to
The Academic Faculty

by

Kyle Williams

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
December 2019

OPEN SOURCE CNC CONTROL WITH DIRECT CAM INTEGRATION AND A DIGITAL TWIN

Approved by:

Dr. Thomas Kurfess, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Christopher Saldana
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Lonnie Love
Energy & Transportation Science Division
Oak Ridge National Laboratory

Date Approved: November 25, 2019

TABLE OF CONTENTS

TABLE OF FIGURES

SUMMARY

High bandwidth internet connectivity and ubiquitous computation are poised to enable automated quality assurance, high efficiency predictive maintenance and an integrated logistic support infrastructure for modern manufacturing. Information technology is in the process of revolutionizing production, as it has revolutionized so many other industries. However, old and new CNC systems alike are unable to fully claim this advantage.

Milling machines are a significant capital investment; it is impractical to regularly replace them; aging systems continue to see use, but are increasingly unable to meet modern demands. These demands include tighter machining tolerances, three and five axis automation, and internet connectivity. On the other hand, modern machines evolved in a niche market with a high price for entry; these systems meet performance demands, but employ obfuscated, proprietary hardware/software systems that stifle free market innovation and offer limited bandwidth communication interfaces. They are often prohibitively expensive as well.

In this body of work, an aging CNC mill is upgraded with a modern electrical power system and an open source firmware/software architecture for control and communication. A digital twin of the machine tool is developed directly in the CAM environment, where toolpaths are generated. Leveraging this open platform, the CAM software is connected directly to the machine tool over the internet, enabling remote monitoring and control. This report presents the engineering behind the system, in the broader context of the need for open source control and the demands on modern machine tools. The system is vetted out on a 1986 Mori Seiki vertical milling station and experimentally verified.

CHAPTER 1 INTRODUCTION

Lean adaptive manufacturing will provide substantial savings in time, efficiency and money in Industry 4.0. Consider the paradigm in anecdotal form. A hardware engineer orders 500 components for life cycle testing from a nearby facility online. The process design is largely automated; AGV's and manipulators load the stock; the mill begins material removal. Continuous process variable feedback and analytics ensure all machines are healthy and all parts are produced within spec. After transitioning through the remaining finishing steps the parts are loaded on autonomous freight and arrive at engineering within hours. The reduction in cycle time, man hours, logistic expense and waste yields lower prices for development and production.

Focusing now on the CNC machine, this vision translates into a set of engineering requirements. First, this is not the rigid production environment of Henry Ford; the machine requires the integration of flexible, large scale software systems. Second, the part should be producible on any CNC with limited human intervention; the machine requires a universal command language and an internet based control interface. Third, the engineer should have a high degree of confidence in the process; the machine requires a digital twin with high bandwidth process feedback for external monitoring. Finally, large scale deployment requires that the machine be economically viable in small scale production environments.

These requirments are illustrated in Figure 1. The modern CNC must be a flexible part of a network of growing complexity. On the shop floor, the CNC must be capable of integrating with new software and new sensors, and of communicating with the robotics and other manufacturing systems at play. All this data must flow to the software layers

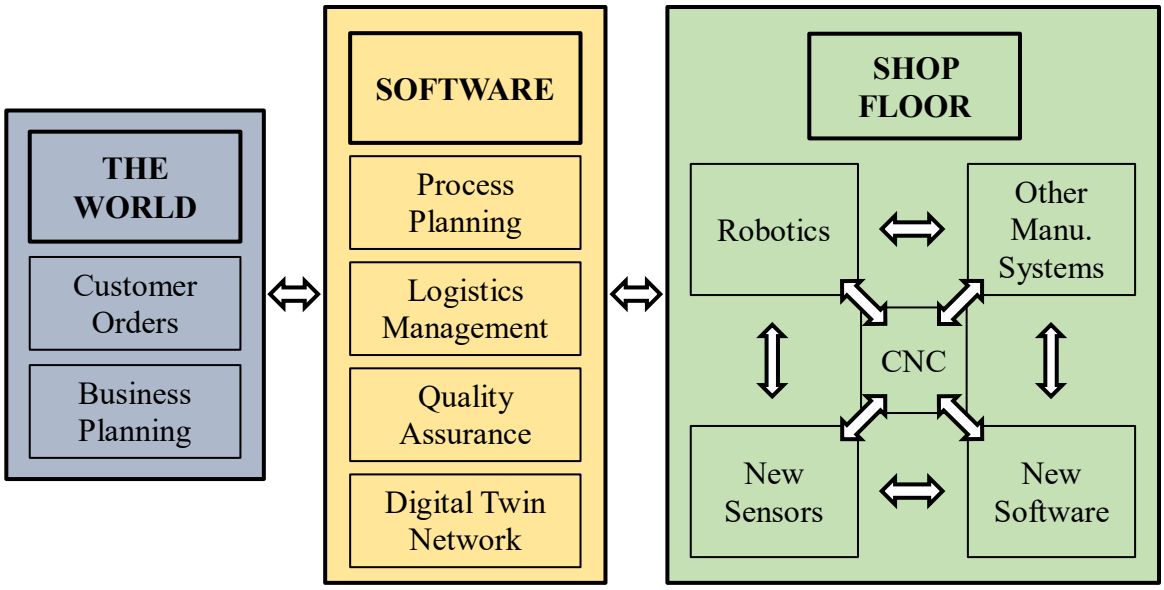

Figure 1. The CNC machine within the cyber-physical network of modern manufacturing

responsible for automated process planning, mangament, and quality assurance, with the high level goal of delivering a higher quality service to the customer at reduced overhead.

The majority of CNC machines in use today do not meet these requirements and are unequipped for Industry 4.0. Modern systems employ monolithic communication and control systems with a very limited interface for flexible hardware or software integration. Each machine runs a unique brand of G Code and requires a human to load the code over USB. There is no process feedback or digital twin for quality assurance.

Finally, these machines are too expensive to enable local, distributed production. The older machines more common in local shops do not meet the requirements either. These machines vary widely in capability, but often cannot meet modern tolerances or mill continuously in three or five axes. The older machines offer no connectivity, no internet interface and no process feedback.

This work proposes an inexpensive hardware update for aging systems, along with an open source computation, communication and control platform offering total software flexibility. The system includes an electrical power system for controlling mechanical systems such as way oil, coolant, an automatic tool changer, and drive axis ball screws. A generic PC running a patched version of Linux controls the hardware, interfaces with the internet, and runs higher level software. Building up, a remote interface is integrated for machine control and process feedback. The system is implemented on a 1986 mill and experimentally verified.

The mill is shown in its original sheet metal enclosure and with its original control interface in Figure 2. Figure 3 shows the original 80's era control system, as well as the stripped down version of the machine. This hardware serves as the platform for the investigation.

Figure 2. Machine hardware (left), control interface (right)

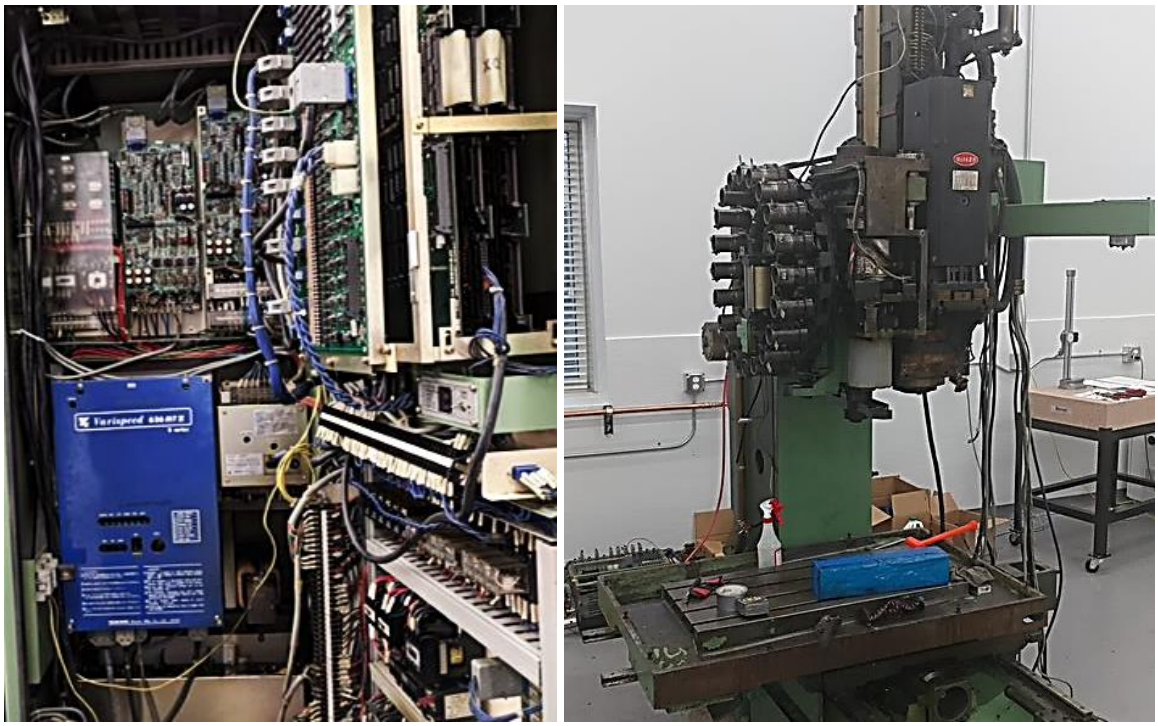

Figure 3. Original control cabinet (left), and stripped down machine (right)

CHAPTER 2 BACKGROUND

Utilizing computation and communication technology for improving production has become a major multinational endeavor among industrialized countries. Policy changes and technological goals have been promoted as Smart Manufacturing in the US and Industry 4.0 in Germany, with similar programs in Korea and Japan [1, 2].

This new industrial revolution is driven by market competition, evolving customer needs, and advancing information technology [3]. Among manufacturers, reduced labor and energy overhead, better service, and increased resource utilization will provide a competitive advantage. Additionally, customers are demanding increased production flexibility, smaller batch sizes and lower costs. Finally, the explosive growth of computing, communication, and automation technology will provide the foundation for these advances, if they can be deployed effectively. A cyber physical system is the integration of computing, networking, and a physical process [4]. The new industrial revolution will utilize increasingly complex cyber physical networks to yield more automated, more efficient, better quality production.

The effort to take this new industrial revolution from conceptual vision to pragmatic implementation has resulted in a significant body of industrial and academic literature. The prior work relevant to this project consists of an aggregate of publications grouped as follows:

- Integration needs and architectural specs
- Open source vs. proprietary systems
- Limitations of G Code and other control types
- Process optimization algorithms and AI

- Monitoring, control and automated QA
- Digital twin concepts and implementations

2.1 Integration Needs

The scope of manufacturing is no longer merely the physical process of changing raw materials into goods. The field is now considered the broader integration of physical processes, with planning and logistics systems and the entire operation of the business [5]. The ANSI/ISA95 specification defines a standard for interfacing between levels of the manufacturing enterprise [6]. Levels 0, 1, and 2 deal with the physical processes and machines, level 3 deals with manufacturing logistics, and level 4 deals with the longer term business functions within a manufacturing company. A detailed description of the specification is not required here; the point is that information needs to flow from the physical manufacturing process to the decision making bodies of the manufacturing organization in a coherent, actionable manner. However, in many production companies, the information at the bottom levels of the pyramid is still entered manually in a sparse and error prone manner [6]. The goal of organized, automated information flow is a large part of the Industry 4.0 vision. Standards are continually under development to facilitate this process.

2.2 Open Source

A primary impediment to the free flow of data is the heterogeneity of obfuscated, proprietary communication protocols, hardware systems, and software packages. In a book on ISA95 implementation, B. Scholten discusses the time consuming and error prone process of integrating control systems and information technology when “confronted with differences in technology, metadata, programming languages, user interfaces, and more”

[6]. This need for interfacing disparate systems has forced development of a wide range of interfacing standards such as OPC UA and MT Connect, along with a number of publications on open source platforms [7, 8]. Vijayaraghavan et al. state, in their publication on MT Connect, that a pipeline from engineering to manufacturing, which integrates all intermediate bubbles of advanced technology, is a long standing vision for the industry [8]. The publication cites communication between the vast numbers of proprietary machines that transform the raw material into the final product as a primary limitation to realizing this vision, and suggests MT Connect as a remedy. In a master's thesis at Clemson University, Pezzulo states that limited access to data in proprietary machine tool controls has limited implementation of otherwise viable automated chatter reduction software, and proposes an MT Connect based solution [9]. However, in that body of work, there was no feedback mechanism to modulate spindle speed upon chatter identification, due to the lack of an internet based or flexible programmable control interface on the proprietary CNC. Despite the widespread research in open source CNC control, such platforms are still not widely available.

Communication protocols like MT Connect and OPC UA offer promising solutions for standardizing information flow between systems, but the principal issue is still the proprietary nature of manufacturing systems. MT Connect and OPC UA only provide access to the data made available by the machine manufacturer. Additionally, process parameters often cannot be changed over the wire, so sensor or software integration is impossible. In short, these communication protocols only solve half the problem; for true agile, flexible, data rich manufacturing, the CNC system nodes in the communication

network need to be open source. They need to allow unconstrained monitoring, control, and software integration.

There have been a number of research efforts dedicated to developing open source CNC systems. The most notable is the Enhanced Machine Controller or EMC developed by the National Institute of Standards and Technology or NIST [10]. The motivation for that project was to enable more rapid innovation and software integration on inexpensive and generic computer systems, and to gain an advantage over the proprietary systems employed by competition outside the US. The publications on EMC cite the possibility of third party software integration, only viable on an open source platform, as a major growth opportunity. Additionally, decoupling hardware from software reduces overall system prices and opens up a new market for entrepreneurial development [11]. The work by NIST is not the only attempt at increasing transparency in machine tool control. In another body of research, Schofield and Wright discuss design principles and an implementation of open source machine tool control [12]. The authors again emphasize the growth potential of incorporating third party software. The authors further emphasize the inflexibility of standard machine tool programming interfaces, and the inability to integrate additional sensing or meaningfully improve machine tool capability. The authors also discuss the economic losses associated with inflexible automation systems that require large upfront investment, but are too rigid to remain valuable when production needs change. In another publication, Koren et al. state that reconfigurable manufacturing on a basis of open architecture controllers will be fundamental to competitive production in the coming decades [13]. As a more recent example, Correa et al. developed an open architecture CNC based on modern, open, single board computing technology [14]. Additionally, nearly all

academic experimentation in CNC machining is performed on open platform CNC machines developed on a lab by lab basis, as this is a requirement for testing novel functionality in today's state of the art. This will be explored further in future sections.

2.3 G Code

The proprietary nature of CNC systems makes them difficult to improve, but there are more specific problems too. For example, there have been several attempts to transplant G Code as the industry standard numerical programming language. After years of development, the G Code specification was released in 1979 with the goal of enabling the execution of simple machining primitives [15]. The standard was developed in an era when processing power and communication bandwidth were orders of magnitude below present standards. There are a number of limitations under G code; they have been most thoroughly articulated by Xu et al. [16]. The standard is decades old and each machine tool manufacturer has added its own desired functionality, thus limiting the universality of the specification and the interoperability between machines working on the same part. Use of the standard also generates information loss, because the part model is reduced to motion commands which are executed on the machine in a unidirectional manner. The preferred system would use a single representation for CAM and CNC and ideally for process planning and design as well, with homogeneous bidirectional information flow between layers. Additionally, the G Code path is fully specified at the start of machining and the machine tool runs the path in an open loop manner and through a slightly different trajectory from machine to machine. The next generation of machine tools will likely utilize significantly more closed loop feedback and optimization associated with the machining process. Research into such optimizations will be discussed further, but the

machine controller should be able to actively modulate spindle speed to reduce chatter or intelligently increase feed-rate when machining a plastic relative to a steel part, to give a few examples. STEP-NC is one of the more recognized alternatives, but it has not yet supplanted G Code, so a detailed discussion of the merits and demerits of the system is omitted here [17]. NCML is another standard intended for internet based machining [18].

2.4 Optimization

The list of academic research projects that require both an open source machine controller and a programming language more flexible than G Code is vast, but a sample will be presented here. Zhang et al. developed a time optimal trajectory generator for manufacturing systems, but implementing the algorithm on a physical machine would require a controller that can run or interface tightly with novel software and grant complete control of the machine tool trajectory to that software; this is not possible under G Code [19]. Timar et al. present a selection of such time optimization algorithms and explicitly state that their continuous nature makes G Code approximations unnecessary and that experimentation requires an open source controller [20]. Similar time optimization research abounds [21-23].

2.5 Monitoring, Control and Q.A.

Another active area of research is automated monitoring, control, and quality assurance. As a high impact example, Sandia National Labs recently finished a three year, thirteen million dollar project that uses in process monitoring and optimization to produce low volume parts in an additive manufacturing context [24]. The parts are said to be born qualified. In another publication, Soucy reviews research in the use of in process monitoring and quality assurance for composite aircraft structures [25]. As another

example, Munasinghe describes a modular architecture for integrating sensing and software on a CNC machine tool with controllable feeds and speeds [26]. Soori et al. released a paper describing a system that monitors tool deflection during machining and uses genetic algorithm based optimization to minimize this source of error [27]. Li et al. proposed a multi stage automated QA system for integrated manufacturing processes [28]. Wang et al. proposed another framework for remote monitoring and control over the internet [29]. These examples should serve to demonstrate interest in the ability to actively monitor process variables and modulate control parameters accordingly.

2.6 **The Digital Twin**

Additional research related to monitoring and quality assurance is focused on the use of digital twins. As a high impact example, NASA and the US Air Force stated they would no longer use exclusively statistical models for predicting reliability, but would increasingly utilize a digital twin which integrates simulation, onboard sensing, maintenance records and historical data [30]. In a manufacturing context, Kritzinger et al. state that digital models, digital shadows, and digital twins provide various means of synchronizing the physical and digital world for integration with the other systems of industry 4.0 [31]. Kannan et al. published a case study where the use of a digital twin increased grinding efficiency by 14% [32]. Soderberg et al. claim the use of digital twins for geometrical quality assurance will contribute to enabling profitability with smaller batch sizes [33]. Similar publications on digital twin implementation is plentiful [34-36]. These examples should serve to introduce and motivate the use of digital twins.

2.7 Background Summary

The concepts presented above are well summarized in a paper by Xu [37]. Xu's paper states that industry 4.0 needs machine tool 4.0. MT 4.0 should be integrated with design tools and with other manufacturing systems; it should not use G Code; it should be constantly monitoring the health of the machine and the quality of the part; it should be optimizing all operations; it should have digital twins representing the machine and the part. The body of work presented here is a case study in integrating these concepts using modern, flexible technology. As will be further discussed in the main body, a 1986 CNC mill is controlled using a Linux PC, open networking, python and C, a modern fieldbus and modern servos. The machine is directly integrated with a CAM system and basic digital twins of the machine and the part. G Code is replaced by point clouds and a time optimal trajectory planner.

CHAPTER 3 METHODS AND DESIGN

This section introduces the subcomponents of the project. The subsystem overview is illustrated in Figure 4. The goal is to develop the software, electrical control hardware, and mechanical interfaces necessary to control the existing mechanical systems.

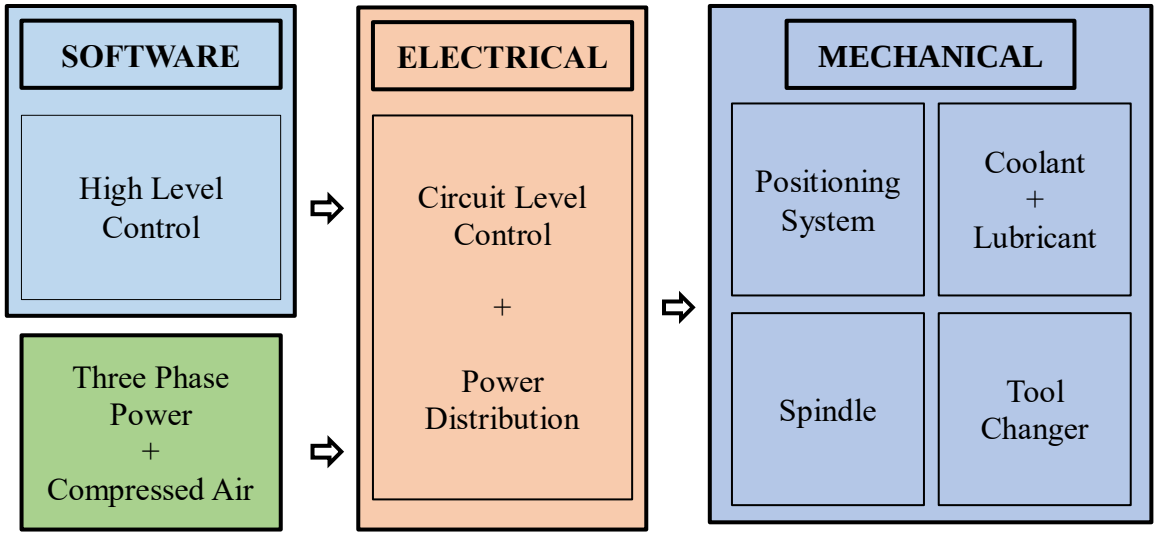

Figure 4. Engineering subsystems overview

The discussion begins with the mechanical subsystems, common to many CNC mills, which guide the design. The subsequent section presents the electrical power systems, followed by the industrial communication network used to orchestrate low level communications and control hardware. The next section outlines the computation architecture employed as the master of machine hardware, the interface with the internet, and the platform for application software. The final sections address higher level communication mechanisms and software structures.

3.1 Mechanical Systems

The upgrade was performed on a 1986 Mori Seiki MV Junior Vertical Milling Center. From a high level, the mechanical subsystems of interest are the three Cartesian drive axes,

the automatic tool changer, the coolant system, and the way oil system. The mechanics are largely in place and reusable, so little mechanical development is necessary; they are described here to motivate the remainder of the design.

3.1.1 Drive System

The drive system consists of three sets of ball screws and linear guide ways. The original motors driving these axes are replaced with state of the art Yaskawa servos. For reference, these are the SGD7S-7R6AA0A model amplifiers, SGM7G-09A7DKS motors on X and Y, and the SGM7G-09A7DKE motor on Z [38, 39]. The primary mechanical points of consideration when selecting replacement servos are the torque/power output, the encoder resolution after accounting for the ball screw ratio, compatibility of mounting

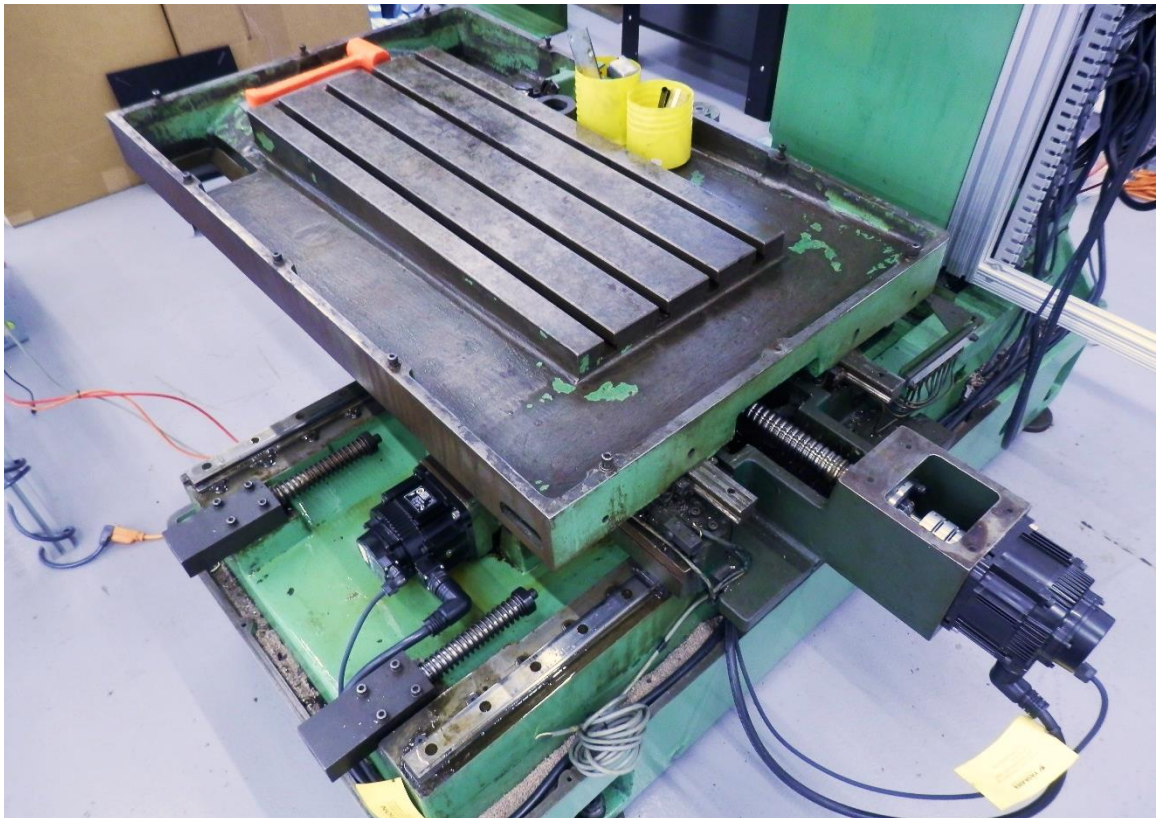

Figure 5. X and Y axis drive system: servos, shaft couplers, ball screws, and linear ways

hardware, and the ball screw to motor shaft interface. The servos used here provide configurable 20 to 24 bit encoders. The pitch on each ball screw is approximately .4” per revolution. At 20 bits, this yields a linear resolution of approximately 10 microns. A good next step would be the implementation of the 24 bit option. The servos also provide 5.39 Nm of torque, which is greater than the outdated servos being replaced and deemed sufficient for that reason. As another note, the shaft interface should be flexible along axes other than the axis of rotation. This intentional point of flexion allows any misalignment between the ball screw and motor shaft to dissipate. Without this degree of freedom misalignment can manifest as stress imbalances in the motor bearings and lead to reduced operational life. The shaft coupler can also serve as a vibration damper which can reduce deviations in machining accuracy [40]. The X and Y axes are driven by semi rigid shaft adapters. The updated X and Y drive system (servos, couplers, screws, and linear ways) is shown in Figure 5.

In this machine, as in many CNC machines, the Z axis motor runs alongside the Z axis ball screw. That arrangement necessitates the use of a belt drive on the Z axis. The assembly provides the desired flexibility as well. Additionally, the Z axis is balanced by a large counterweight attached by sprocket and chain over the high point of the machine. The Z axis servo has an electronic brake as a supplemental measure against gravity. The newly installed Z axis servo and belt drive, along with the couterweight, are shown in Figure 6.

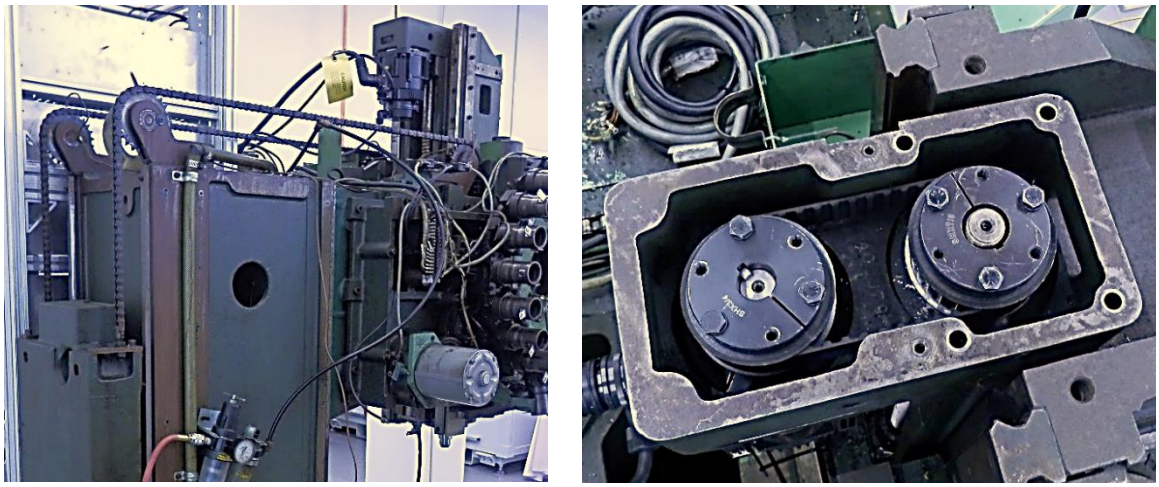

Figure 6. Z axis drive system and counterweight (left), Z axis servo to ball screw belt drive (right)

3.1.2 Automatic Tool Changer

The next mechanical system to consider is the automatic tool changer. Tools are selected from the rotating tool magazine. An induction motor drives the magazine wheel, while a Geneva mechanism discretizes the continuous motion. The passage of tools is monitored with limit switch counters. When the desired tool is in the appropriate position, one pneumatic cylinder is fired to select the new tool while another is fired to disengage the CAT40 pull stud of the old tool from the spindle. A second induction motor drives a cam follower mechanism, which generates the periodic motion of the tool changer arm performing the switch. These pneumatics, hydraulics, induction motors and mechanisms are shown in Figure 7.

Pressurized shop air flows through a pressure regulator through a condensation catch through a flow monitor to the pair of actuated cylinders. The pressure regulator and condensation catch are shown in Figure 8, along with the hydraulic oil resevoir and solenoid valves used for actuating the pull stud mechanism.

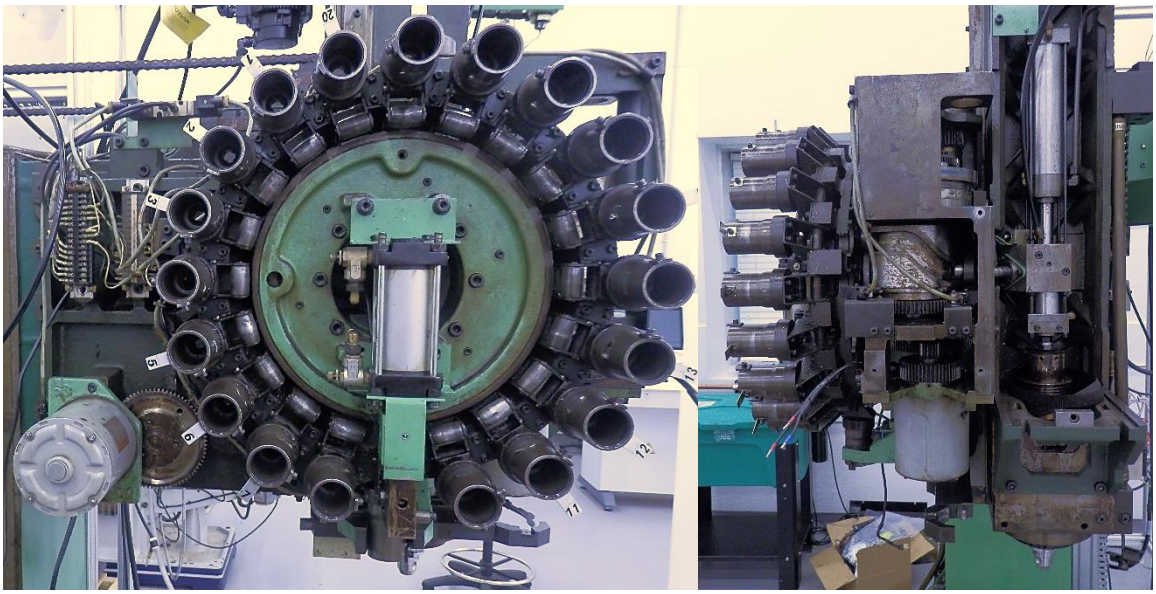

Figure 7. Tool magazine with tool selection piston and induction motor (left), tool replacement arm, with cam mechanism, induction motor and tool holder hydraulics (right)

Figure 8. Pressure regulation valve and condensation catch (left), hydraulic oil and solenoid valves (right)

3.1.3 Coolant and Way Oil

The coolant system consists of an induction motor driving an impeller pump. The pump forces coolant from its tank through a hose to a nozzle in the cutting area. A sloped return path through the cutting bed redirects the coolant to the return tank. This impeller is shown in Figure 9.

The coolant system consists of an induction motor driving an impeller pump. The pump forces oil through a system of copper pipes to the various drive axes in the system, namely the ball screws, their ways, and the chain and sprocket on the Z axis counterweight. This pump is also shown in Figure 9.

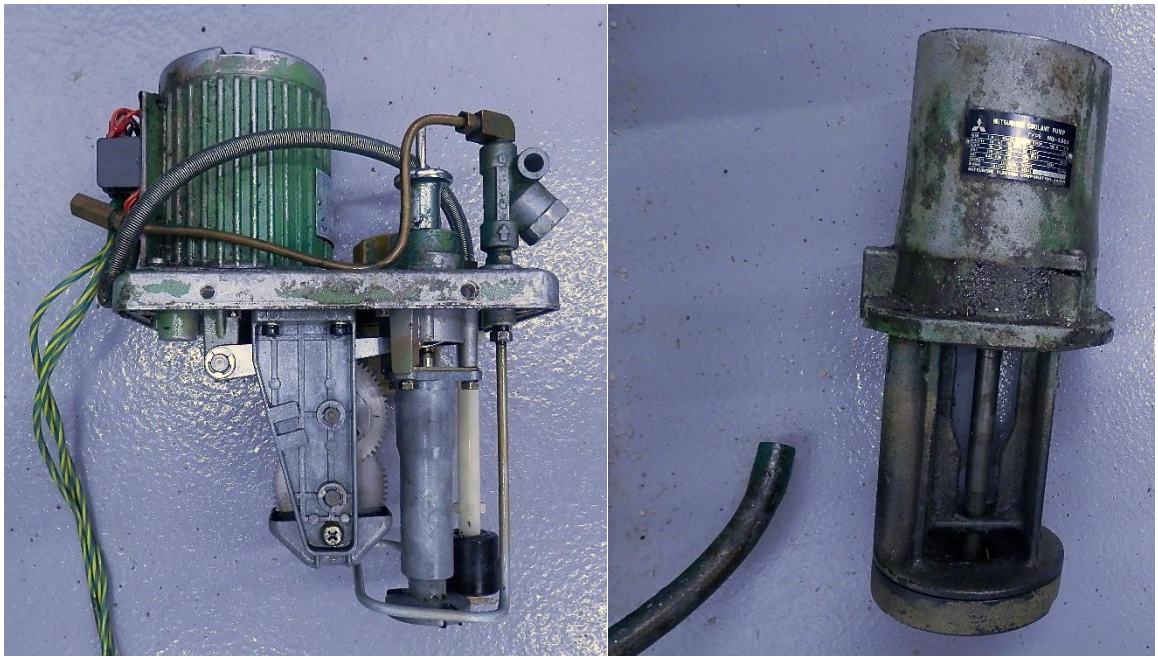

Figure 9. Way oil lubricant pump (left), coolant pump (right)

3.1.4 Enclosure

The primary element of original mechanical design was the enclosure. The enclosure consists of an 8020 frame bolted to the casted machine body, with seven

aluminum plates for mounting the electrical components and din rail. The design includes a network of wire ducting and a simple polycarbonate door for safety.

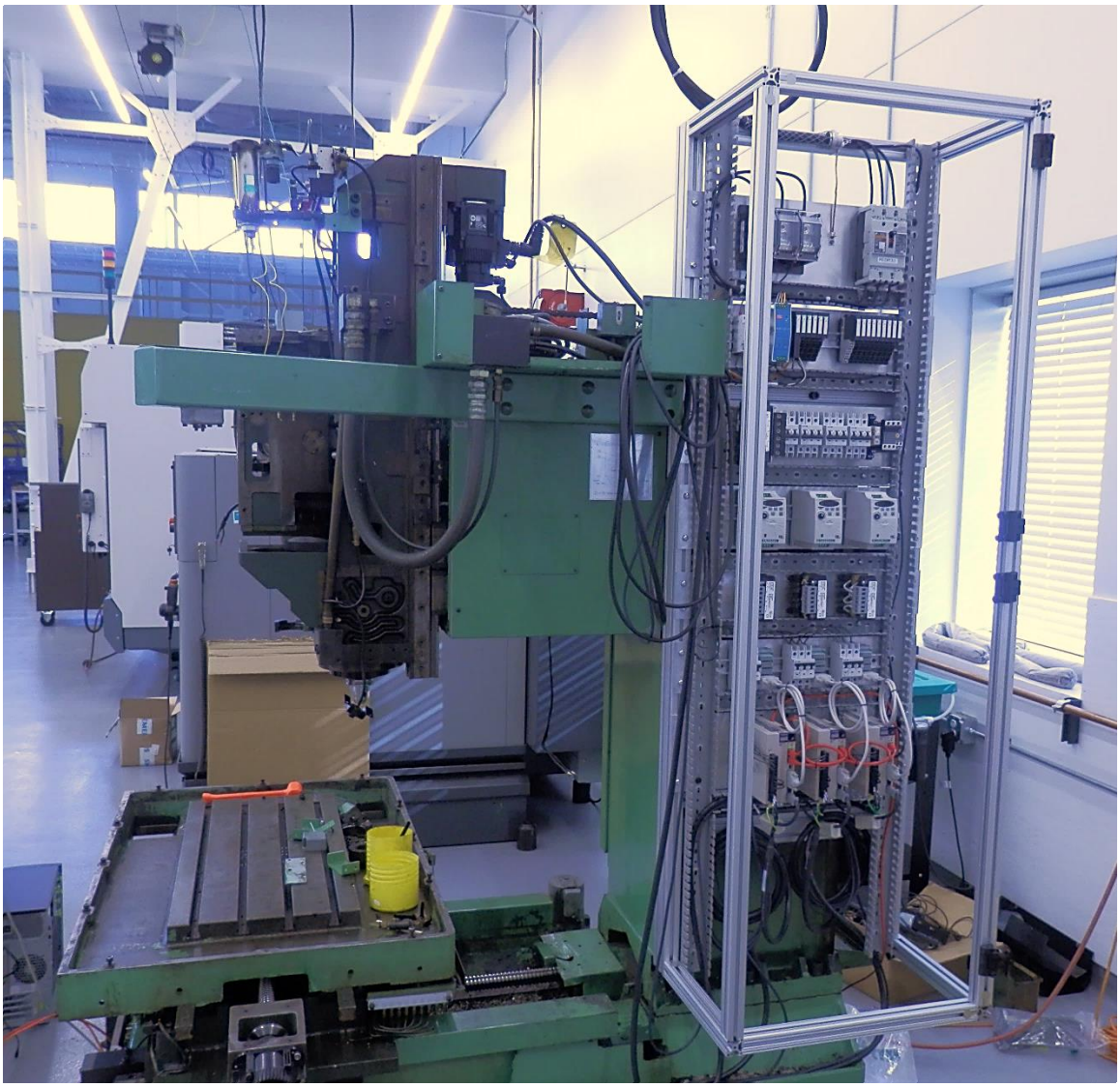

Figure 10. Electrical enclosure

3.2 **Electrical Power System**

The properties of the existing mechanical and electrical systems translate into a set of requirements for the electrical redesign. The drive system demands a means of controlling the 200V three phase Yaskawa servos, releasing the 24V Z Axis brake, and reading various wide range limit switches. The tool changer requires a method of

controlling two 220V three phase induction motors, firing the 24V pneumatic solenoid valves, and reading more limit switches. The pair of 220V three phase induction motors on the coolant and way oil pumps require driving as well. All systems should be safety controllable with a minimum number of power transformations, and with a single communication protocol.

3.2.1 Main Power

The main power system is shown in Figure 11. Beginning with power input, a 220V three phase mainline is dropped in from the shop floor ceiling and tied to the main three phase 50A breaker; the power line ground is tied to the chassis. 50A is the aggregate maximum current consumption of all downstream components plus a 20% safety factor. The main branch breaker provides short circuit protection via electromagnetic induction and slow release overcurrent protection via thermal expansion [41]. The output of the

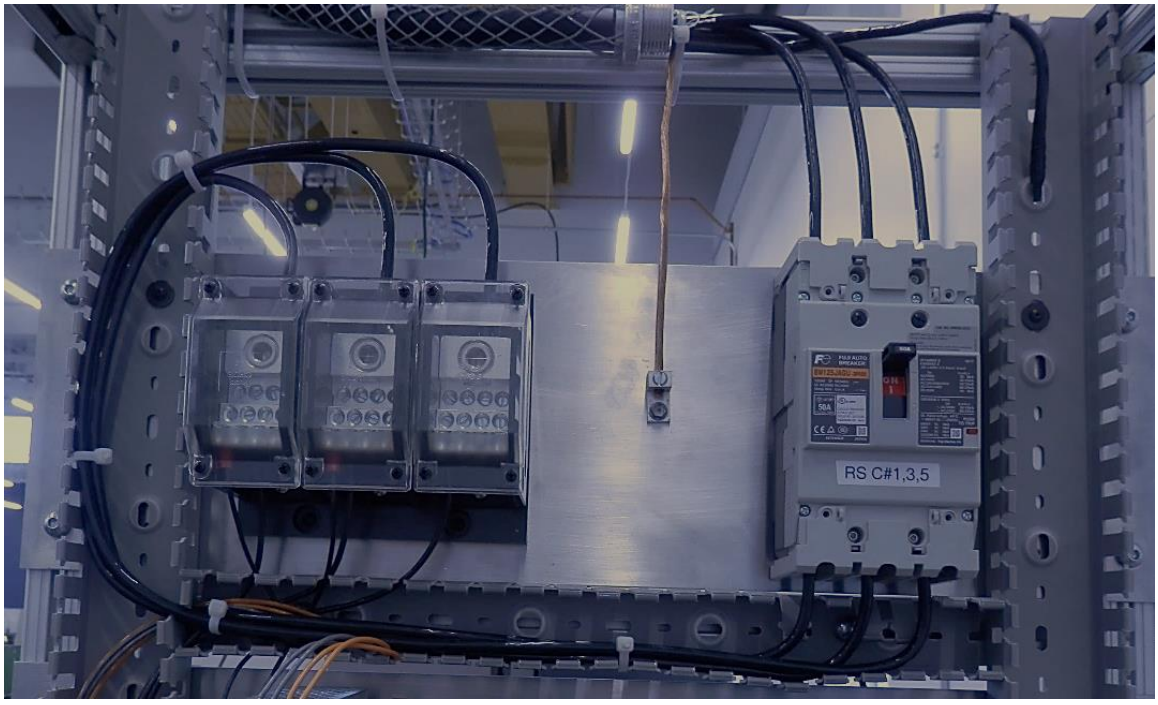

Figure 11. Main three phase power input breaker, chassis ground and power distribution block

breaker connects to three terminal blocks with twelve channels each. This provides power distribution to the rest of the machine. Power then branches into the VFD system, the way oil system, the IO system, and the primary servo control system.

3.2.2 Variable Frequency Drives

The first subsystem to consider employs Variable Frequency Drives to control the induction motors in the tool changer and the coolant pump. These VFDs and their supporting electrical equipment are shown in Figure 12. Induction motors take in alternating current and rotate at a speed proportional to the frequency of oscillation of the supply line; this is 60Hz by US standard. A contactor simply opens and closes three phase terminals; it may be used to drive an induction motor at fixed speed using the 60 Hz standard line frequency. Alternatively, a VFD can modulate this drive frequency. The VFD takes in AC and converts it to DC with a full bridge rectifier and a low pass filter. The VFD then modulates the DC voltage with fast switching power MOSFETS or IGBTs to produce a variable frequency sine wave in the low frequency components of the pulse width modulated square wave. The motor inductance and inertia act as a low pass filter for this square wave and the motor is driven at variable speed.

This design uses VFDs to control the induction motors in the tool changer and coolant pump. This will enable faster tool change cycles and increased control over pump pressure and coolant flow. Additionally, variable speed control enables s-ramped, jerk limited, velocity profiles which reduce stresses on mechanical systems and reduce acoustic noise. However, the comprehensive implementation of tool changer and coolant system firmware is left to future work.

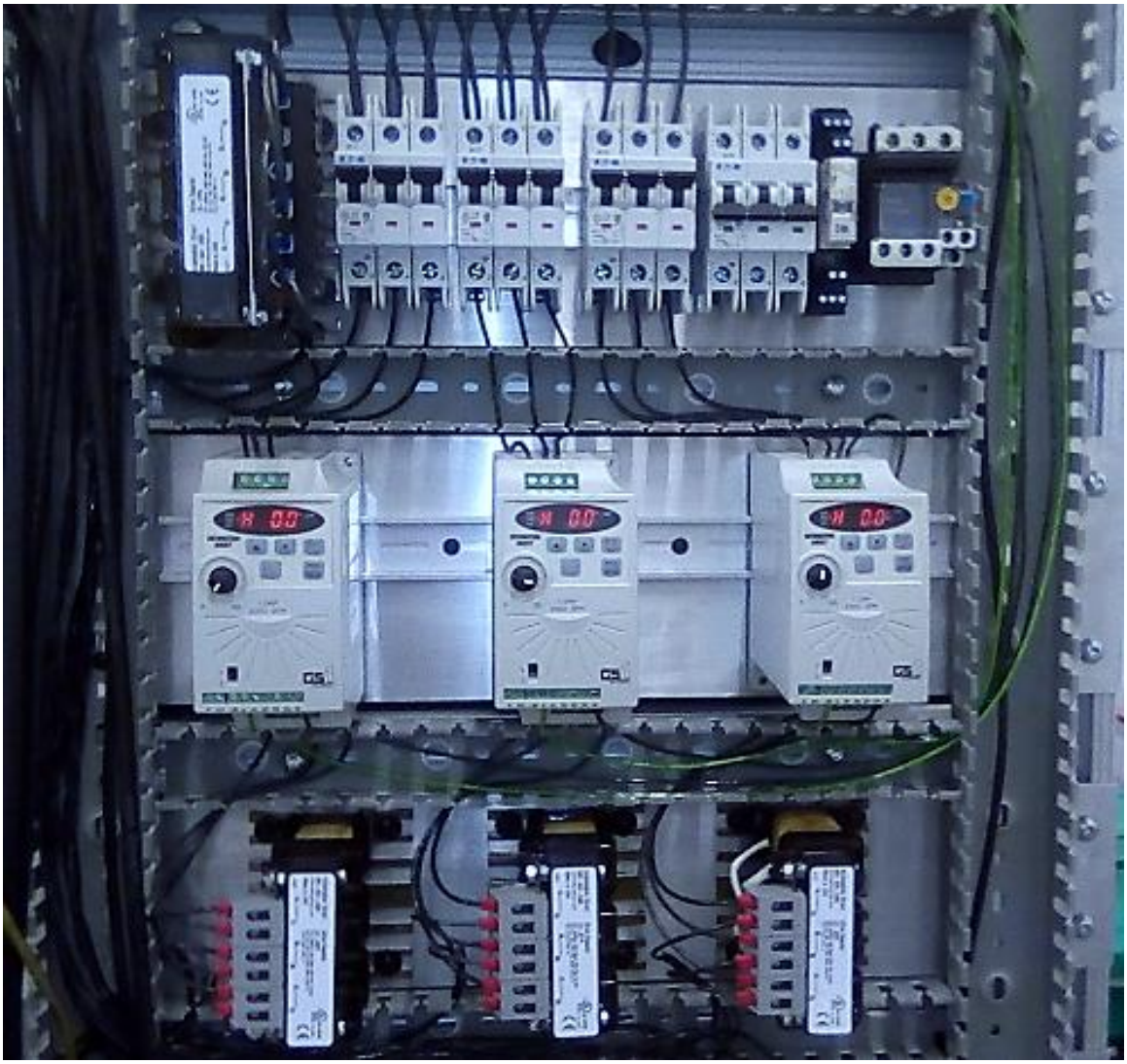

Figure 12. VFDs, breakers, current filtering, and way oil control components

Despite the benefits of VFDs, their introduction creates the need for current filtering. High frequency switching can cause current ripples to propagate away from the VFD both upstream and downstream. These ripples can affect the power factor or stability of other components, both inside and outside the system. For this reason, a 3% reactor is placed in line before the VFDs. This isolates the high frequency signal within the branch [42]. This is not necessary in the case of the Yaskawa drives, as they have built in chokes on this effect, but the problem is always present. The reactor is essentially an inductor that

functions to limit the rate of change of current to an acceptably low level. 3% is a standard size for this application. Next in line comes a breaker prior to each VFD and the VFDs themselves. After the VFD, there is an additional load side reactor between the drives and the motors. The load side reactors serve as a protective low pass filter on the high frequency output of the VFDs. The induction motors used for the tool changer and coolant system are from the original 1986 machine, and rated for constant velocity operation. This means the insulation class on the windings is not sufficient to withstand high frequency content for extended periods of time. Failing to filter out this content before driving motors of this class can cause thermal fatigue and reduce the life of the motor.

3.2.3 Way Oil

The next subsystem is the way oil drive system. Three phase power from the power distribution block is brought to a branch breaker and then to a contactor rated for a 220 V supply, 5A max motor current, and 24V digital IO control. The line is then passed through a thermal overload relay (TOR) and out of the enclosure to the way oil pump. In contrast, the VFDs and Yaskawa servos have built in thermal overload. The standard way oil pump has an adjustable volume per unit time output so a VFD was not necessary to drive this motor. The contactor and thermal overload are shown in the top right corner of Figure 12.

3.2.4 PASIO

Continuing with the IO system, two legs of 220 V alternating current are brought to an appropriately sized breaker for this branch. The breaker outputs to a 24V by 5A DC power supply. The DC power supply drives the logic and power rails of the power module on a Power Automation Slice IO (PASIO) analogue\digital input output system [43]. The rack includes an extensible number of ports. The ports include high amperage digital IO

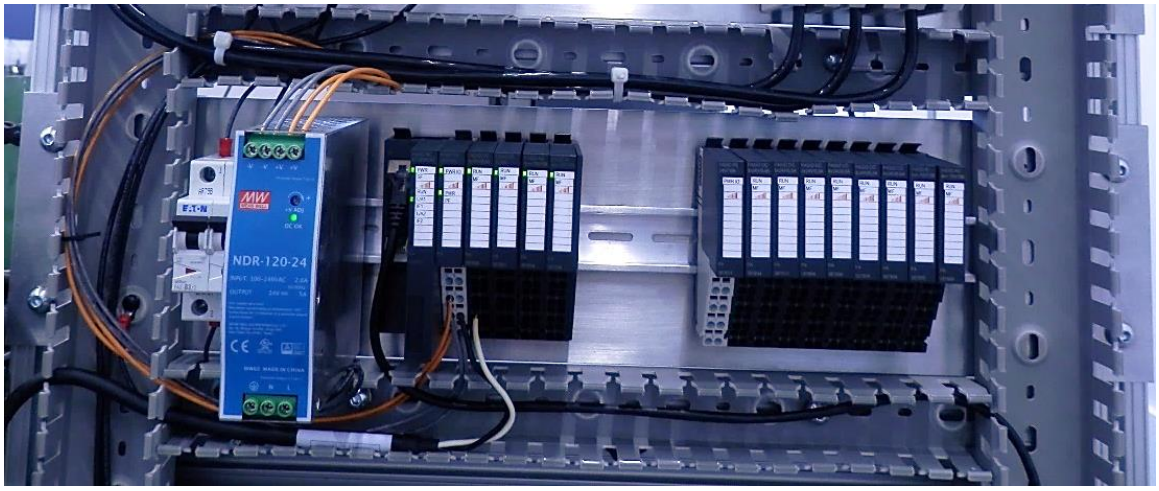

Figure 13. Single phase DC power supply with breaker and IO Rack

for controlling the tool changer solenoid valves, Z axis brake, and way oil contactor. The ports also include analogue output capable of controlling speed in the VFDs. The ports have digital inputs for reading the various limit switches across the machine and analogue to digital converters for reading any analogue sensors that may be added. The IO rack, DC power supply and two line breaker are shown in Figure 13.

3.2.5 Servopacks

The most important electrical block is the Yaskawa servopack subsystem, shown in Figure 14. These servopacks are designed to drive the required motors with the required power level and the preferred communication method (to be discussed later). Each servopack is preceded by a branch breaker connected with the power distribution block. Each servopack takes three phase drive power and single phase logic power from its breaker. Each servopack outputs three phase power and a ground line to the motor, while reading back a 20 to 24 bit encoder value. Each servo also connects with a 27 pin din rail mounted breakout board. This board can read limit switches, temperature sensors, or

general IO and respond in a programmable way. However, exploring this functionality is left to future work.

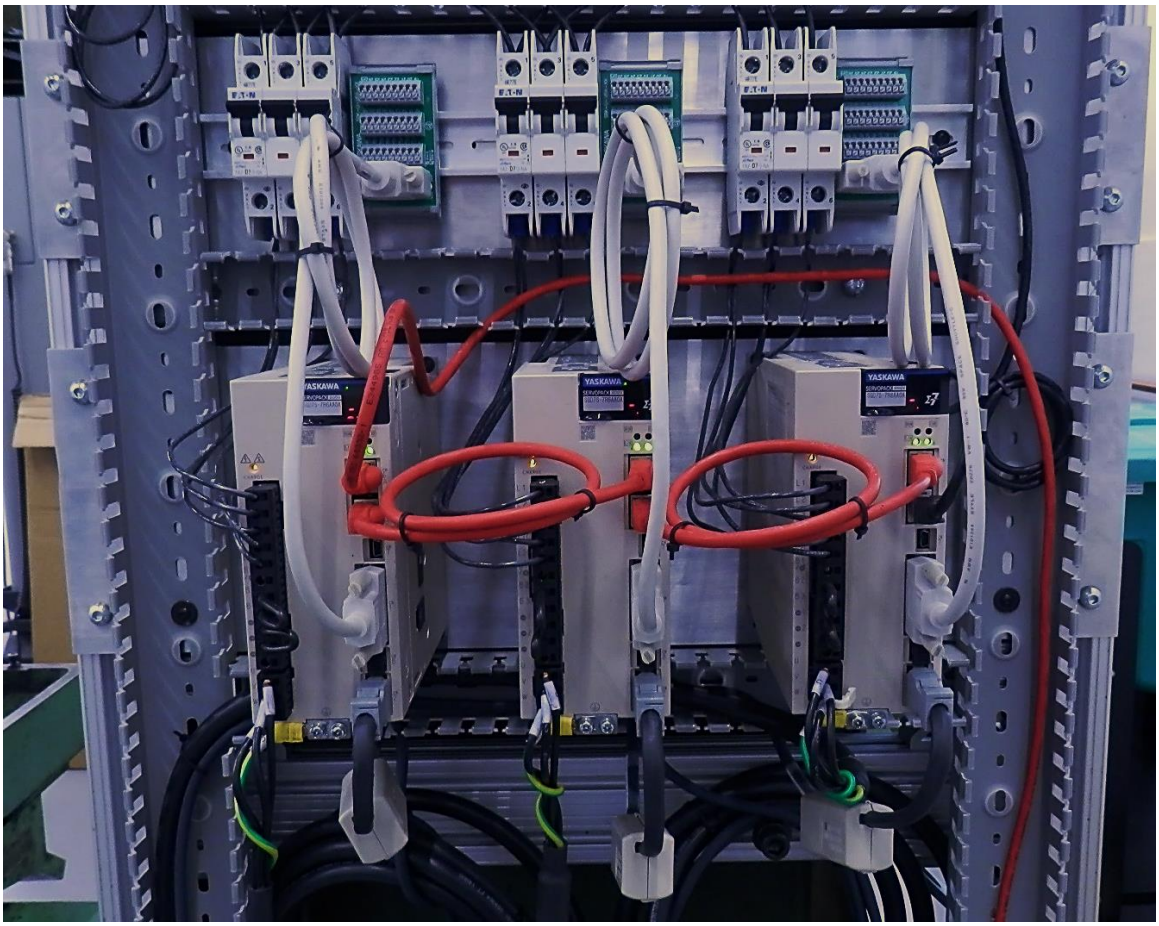

Figure 14. Yaskawa servopacks, with breakers, breakout boards, motor leads, encoder leads, and communication lines

3.2.6 Servo Control System

The job of a CNC machine is to trace desired reference trajectories with a cutting tool. This is done layer by layer until stock material is converted into a final part. Where these toolpaths come from is covered in future sections. The present concern is the feedback system the machine uses to execute these desired trajectories. One of the primary goals of developing an open source CNC machine is the integration of multiple layers of feedback control. This system could be used to test different sensors and more advanced

control algorithms such as zero phase error tracking control, especially in the developing field of additive manufacturing. However, the current implementation performs all feedback control directly on the servos. A high level block diagram of this system is shown in Figure 15. Position reference command updates come in at the update rate of the control system, to be described later. This period is one millisecond. The first feedback loop uses a Proportional Integral (PI) controller, with anti-windup, on position error. This is fed to another PI controller on velocity. A portion of the original position reference signal is differentiated and fed-forward directly to the velocity control loop, to reduce tracking error in the position loop. Without a feedforward term there must necessarily be error in the position loop to generate motor torque. In general, increasing the feedforward term reduced mean tracking error, but tended to increase oscillatory behavior, leading to acoustic resonance in the extreme case. The output of the velocity control unit is fed to a torque filter block. The position and velocity loops have no damping terms, so damping behavior of the total system is tuned by increasing or decreasing the time constant on this torque filter. The block also includes optional notch filters for reducing excitations of the resonant

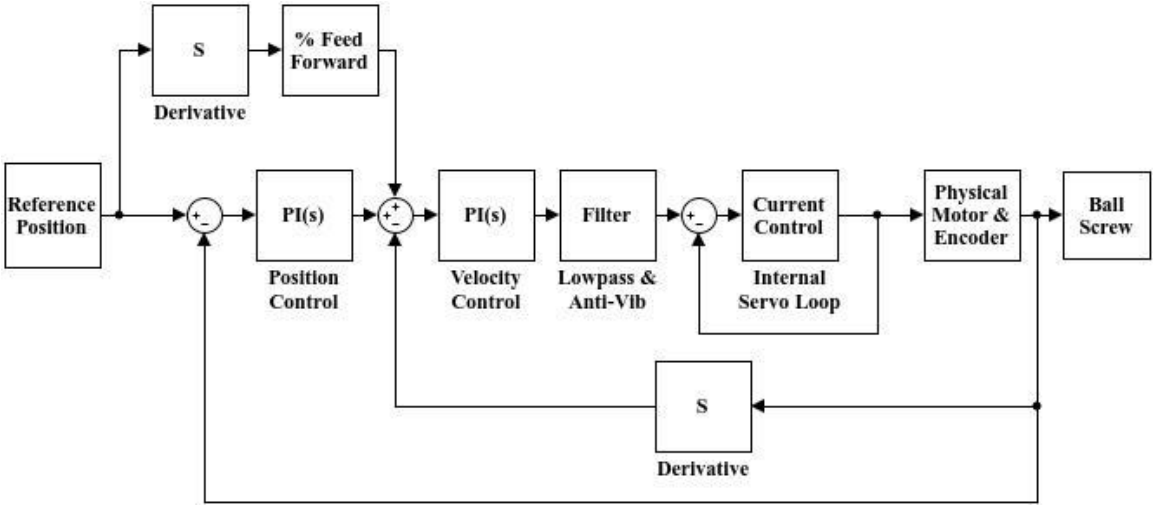

Figure 15. Simplified block diagram of the servo control system

frequencies of the rigid mechanical system driven by the servo. The servo then feeds torque demand into its internal current control loop. Additionally, the servos come with an auto tune feature for initial parameter setting and moment of inertia estimation on each axis. The gains can then be tuned manually to achieve desired performance. The performance of this control system will be evaluated in the results section.

3.3 CANopen over EtherCAT communications

All the hardware in the above electrical system is controlled via CANopen over EtherCAT (CoE). CANopen (open Controller Area Network) is a communication protocol; EtherCAT (Ethernet for Control Automation Technology) is a fieldbus specification [44-46]. In terms of the OSI network stack, CANopen approximately covers the network, transport, and application layers, while EtherCAT approximately covers the physical and data link layers. The primary benefits of CANopen over EtherCAT are its ubiquity, flexibility and its optimizations for real time, distributed processing. These features accord nicely with the overall project goals of clean, high fidelity, real time monitoring and control.

The ubiquity of the CoE interface reduces the cost and complexity of the hardware upgrade and higher level software integration. Automation hardware supporting CoE is easy to find. The Yaskawa servos and the Power Automation IO rack run this stack, and the IO rack controls the tool changer, coolant and lubricant systems. This means the entire electrical hardware system uses the same protocol. This eliminates the need for communication adapters or multimodal communication hierarchies. This also brings all mechanical state data into one network and one data structure that can easily be delivered to higher level applications. This enables tight integration of high level software and

networking with low level control. The overall effect is reduced cost, size, and development burden.

EtherCAT is flexible. Each device is full duplex and nodes can be chained together in a multitude of configurations. This flexibility lends itself to a general purpose machine update architecture.

EtherCAT is well suited to low latency, distributed real time processing. When in the operational state, the fieldbus sends a single datagram through the entire local area network of machine components. This repeats at a fixed time interval, which is 1 millisecond in this case. Each slave device extracts incoming data off the bus and places outgoing data on the bus in real time, as the datagram passes through the hardware. A preconfigured addressing scheme simplifies networking and reduces the need for additional packaging bits on each datagram. This reduces the minimum cycle time of the system. Altogether, this yields a communication system with minimal wasted overhead and low latency.

The EtherCAT system employs a distributed clock mechanism that aids real time performance. Each node in a distributed network runs an independent processor. If these processors clock at 1 GHz, and neighboring processors are manufactured to one in a billion clock cycle precision, the nodes will be out of sync within a second. That is an extreme example, but it illustrates a problem in distributed networks where synchronous communication is a requirement. In this case, any drift in time synchronization directly translates to reduced manufacturing tolerances and reduced system value. This problem is often solved by clocking all communication devices in a network with a single real time clock, either in the master or a specialized external component. For example, in Serial

Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) systems all devices are clocked with a signal from the network master. This requires additional wiring and an accurate real time clock on the master. Alternatively, EtherCAT's distributed clock mechanism passes timing misalignment information along in each datagram; all EtherCAT slaves have an accurate real time clock; at every cycle the entire network is resynced with one such slave. This reduces jitter, which enables reduced cycle times, and limits hardware requirements. For this reason among others, an EtherCAT master can be implemented on the network interface of any PC with an Ethernet port, while the fieldbus protocol, rather than specialized hardware, handles timing. This is explored in the next section.

The final benefit of CoE is low cost. The fieldbus runs on the Ethernet physical layer with Ethernet network interfaces. This technology is already in mass production and widely available. This enables a generic master, cheaper slave devices and the use of inexpensive CAT5 cabling. The CAT5 cabling connecting the servos can be seen in red in Figure 14, in the servo section. The cabling also connects to the IO rack, and is seen in black in Figure 13. Finally, the cabling and the network connects to the central control computer to be discussed next. It can be seen exiting the rear of the computer in red in Figure 16.

3.4 **Central Computation**

The machine controller must serve three purposes in order to achieve the flexibility, communication and integration goals laid out in the introduction. The controller must serve as a master of the CoE network, support unrestrained software integration, and support high bandwidth internet connectivity. These requirements should be achieved in a way that lends itself to general deployment on a variety of systems at low cost. This is accomplished

with a generic PC running a Linux kernel, a combination of open source kernel patches, and the addition of a network interface.

3.4.1 Hardware and OS

The control system employed is somewhat abstracted from computer hardware, but in this case, the PC runs on an Intel Pentium Quad Core processor at 2.5 GHz with 4 GB of RAM, shown in Figure 16. The system runs the Debian variant of Linux. The operating system, kernel patches, and drivers employed here could be installed on a wide variety of systems with diverse processors and Ethernet chips. Indeed, for a portion of the development phase this system was partially incarnate on a BeagleBone Black. Verification and testing is left to future work, but an assortment of inexpensive single board

Figure 16. Control PC

computers could likely run this system. The software components have the added benefit of being free.

3.4.2 Xenomai Real Time Patch

A Xenomai real time patch is added to the Linux kernel to enable precise tracking and feedback of synchronized machine tool paths, velocities and accelerations [47]. The stated purpose of the Xenomai development team is to aid in the replacement of proprietary real time stacks, like those found in industrial controllers, with open source Linux. The Xenomai patch adds a small real time co kernel that runs alongside the native Linux. The native Linux optimizes non-real time scheduling for processing throughput rather than timing consistency. This new kernel has a higher priority than the native Linux and orchestrates all real time tasks such as interrupt handling and real time thread management. The new kernel demands certain blocks of CPU time with very low latency in a nonnegotiable fashion. All non-real time tasks are forwarded to standard Linux for regular management. The Xenomai extension also provides an API for programmable configuration of real time tasks. The API is linked directly to the new kernel scheduler. This API gives direct access to the real time components of the machine controller and increases the software integration flexibility of the system in accordance with the overall goals of the project.

3.4.3 IgH EtherCAT Master Patch

The IgH EtherCAT Master bundle is patched into the Linux kernel to enable the PC to function as a master of the CoE network [48]. The patch uses a portion of the resources of the kernel, processor, memory and Ethernet interface to function as a dedicated master capable of driving an EtherCAT network. The system provides a command line tool

and an API for one off and programmatic interactions with slave devices in the network. This feature increases software flexibility once again.

3.4.4 Network Interface

An additional Ethernet port is integrated to provide the controller with regular internet access, as the original is now a full time hardware control device. This can be accomplished with a USB–to–Ethernet Peripheral Component Interconnect (PCI) adapter, or through the PCI slots on the motherboard of most PCs. The USB approach has the added benefit that it can be implemented on any single board computer with a spare USB port. A USB interface is common on most low cost computing devices, while the PCI interface is not. For that reason and to keep the architecture generic and amenable to low cost implementations, the USB approach is used here. There is a set of Linux drivers that accompany these adapters which must be installed for the interface to function properly.

Two Ethernet cables in red and orange can be seen exiting the rear of the control PC in Figure 16. These are the CoE fieldbus link and additional Ethernet port respectively.

3.5 **Low Level Firmware**

The electrical systems are now primed to control the mechanical systems, with distributed communication and a central controller in place. The subsequent step is firmware design. The following sections describe the firmware required to configure the CoE network, establish the real time loop, and drive the Yaskawa servos. These sections are intended mainly as a service to the researchers taking over the project, and can be skimmed without loss of information on the overall system. The explanations that follow are generally more relevant when viewed in conjunction with the code running the machine.

3.5.1 Network Configuration

The CoE system is optimized for high speed, cyclic communication in a fixed topology network with minimal routing overhead. These benefits come at the cost of additional configuration requirements. The first firmware subsystem establishes this configuration. The first step is requesting a master object from the kernel. The next is requesting domain objects. These domains divide the structure of data flow. There is one for the PASIO rack, one for servo commands and one for servo feedback. The next step is to request slave configuration objects for each device ID at each position in the network. The subsequent steps establish the data in each cyclic datagram or process data object (PDO). A list of relevant memory locations and the size of the data contained there is setup for each slave. These locations are mapped to positions within PDO objects. Finally, these PDO's are grouped into synchronous transmission units under a set of sync managers. This information is summarized in a set of data structures for each device and passed to the kernel. In the following step, these PDO's are linked with the domains previously mentioned. For each device variable of interest, a pointer to the data structures where the information will be stored within the application is established. Each pointer is associated with a device position, device ID and device memory location. A list of these associations is summarized in another data structure for each domain and passed to the kernel. Essentially, all this configuration establishes where the kernel can find relevant memory locations for exchanging data with the autonomous CoE network, and what that data will be. With everything configured, the remaining steps are to request that the kernel activate the master, and to request access to the memory locations where domain data will be stored and updated. The network is now ready for consistent transmission of state information

and commands. These steps are further elaborated in the IgH documentation previously cited.

3.5.2 Real Time Configuration

The next step in firmware is the setup of a fixed time interval loop for PDO exchange through the real time kernel. First, a skeleton function for program logic must be established. At minimum the logic should contain a loop where PDO's are received and transmitted at start and finish respectively. Additionally, a wait flag should be set within the loop to designate the point at which the process must halt between subsequent execution triggers. Next, a request must be made to the real time kernel for a real time task object that references the previously defined skeleton code. Parameters such as task priority, which should be high, and time interval between executions, which should be as low as possible while avoiding overrun, must also be established. The skeleton code can now be filled in with the actual programs needed. Finally, a request is made to launch the real time task. At this point process data flows until the task is terminated. More details can be found in the xenomai documentation previously cited.

3.5.3 Slave Configuration

The CoE network is now configured and transmitting process data at regular intervals, so attention can be paid to the content of the transmissions. The two slave types on this network are the PASIO IO rack and the Yaskawa servos. The simplicity of the former makes it largely plug and play, while the Yaskawa servos require further management. For the servos, there are two relative state machines to consider. There is the EtherCAT state machine and the CAN in Automation (CiA402) state machine. The former consists of transitions between the Init, Pre-Op, Safe-Op, and Operational network states.

This state machine should handle itself automatically if the configuration described in previous sections was successful. CiA402 refers to a standard specifically for servos. The transitions in this state machine are managed by process data exchanges employing the CiA402 control word for commands and status word for feedback. When powered on, the servos enter a “not-ready” state, which automatically transitions to a “disabled” state. The “shutdown” command must be issued to push the servo into a “ready” state. This is followed by a “switch on” command to initiate the “on” state and an “enable” command to initiate the “operation enabled” state. The state machine specifies additional transitions, but this is the common programmatic cycle required to begin operation. Once again, the interested reader is referred to the original CiA402 documentation, which is easily accessible online and in the Yaskawa servopack manual.

The servos are now ready for operation, but the mode of operation must still be selected. The servos offer a wide array of options, but the modes implemented in this system are cyclic sync position, s-curve velocity profile generation, and s curve position profile generation. These are used for toolpath tracking, jogging, and digital read out type positioning respectively. A variety of drive parameters can also be set programmatically or via the command line tool offered by the IgH Master. For example, max profile velocity defaults to zero for safety reasons, and it is necessary to reset this parameter prior to jogging the machine. There are parameters for acceleration and jerk limits on positioning profiles as well. Additionally, there is a parameter for acceptable tracking error beyond which a flag is thrown and the process is stopped. As a final example, there is a parameter that sets the home position on the servo’s absolute encoder. In short, the servo configuration process is extensive and the manual previously cited should be consulted regularly.

3.5.4 GUI

The system is now approaching usability and is ready for higher level software integration. The first block of code to implement is a lightweight GUI. This is a logical first requirement to bring the machine from the design phase into the test phase. The subsequent paragraphs describe the functionality of the GUI and some of the software mechanisms employed by it. As an example of the open and flexible nature of the system, the python tkinter module was employed for developing the GUI [49].

The GUI offers basic DRO (Digital Read Out) functionality. The user can jog or position each axis. When jogging, a go command is sent on button click, and a stop command is sent on button release. This event based functionality is critical for things like incrementally zeroing a work piece. Screenshots of the GUI in position mode and jog mode are displayed in Figure 17 and Figure 18, respectively. The position and speed values displayed are in encoder counts and encoder counts per second. More intuitive units will be used when an accurate measurement of the ball screw pitch is obtained using an

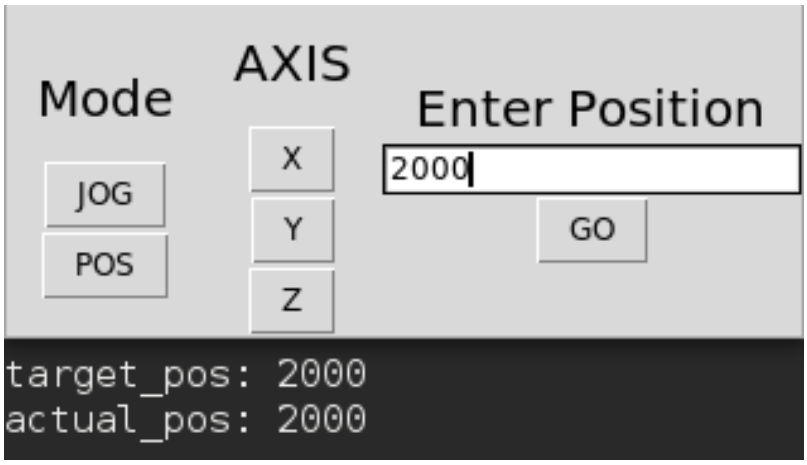

Figure 17. GUI in position mode with zero steady state position error

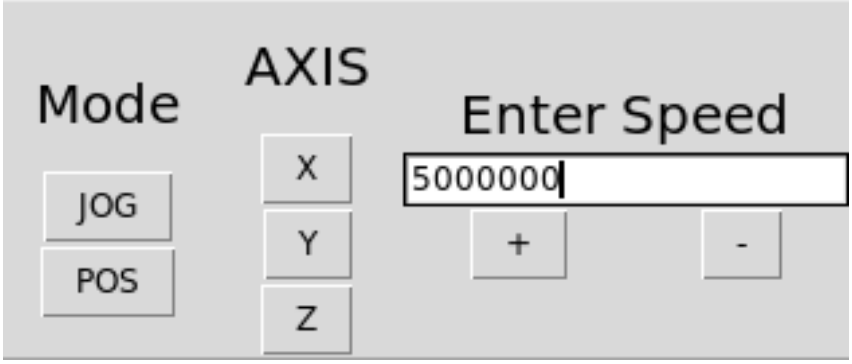

Figure 18. GUI in jog mode

interferometer or some other external position sensor. The 0.4” per revolution number listed in the mechanical section is only an estimate and needs a more precise verification.

The GUI operates through a set of callback functions that modify the parameters of a command data structure. The parameters include the axis, the mode, and the set point. When ready for execution, the data structure is serialized and pushed to the loopback socket, which will be described further next. There is a corresponding socket and parser in the machine control code written in c. The instructions are interpreted by the c code and translated into hardware level setting changes and commands. The GUI is written in python, as opposed to the c used for firmware, because networking and user interfacing comes with a substantially lower line count in python.

A socket is a software mechanism for inter process communication and is the backbone of the internet. Typical sockets communicate data through the network stack between processes running on separate computers. The loopback socket transmits data through the kernel between processes on the same machine. In either case, the two processes hold a file handle from which data can be read and to which data can be written.

3.6 High Level Application

After completing the work described in previous sections, basic machine functionality is in place. In accordance with the goals set in the introduction, the system is inexpensive and flexible with all the internet connectivity of any PC. The following sections show the power of the platform by incorporating software to run an open source time optimal path parametrization module, drop G Code and run tool paths directly from an integrated CAM environment, and drive a digital twin of the machine with encoder feedback from toolpath execution. First, the components will be introduced. Second, their connections will be explained.

3.6.1 TOPPRA

TOPPRA is a method for converting geometric paths into actuator velocity commands. It was previously used for machine control within this lab, in work by Dr. Lynn [50]. The next few sections describe the need for the algorithm, describe the current implementation, and discuss integration with the rest of the machine.

This section describes the need for time parametrization. The GUI described previously allows simple control of individual machine axes, but for machining any complex geometry, all axes must move in concert with precise timing. This requires the desired geometric path to be re parametrized as a function of time. These functions must then be sampled at a constant time interval such that the path can be commanded at the update rate of the servo network and controlled at a hardware level.

Now that the need has been established, the implementation can be discussed. This critical component is another portion of machine control left obfuscated by modern proprietary systems, but in keeping with the open source nature of this project, the system

employs an open source python module. The module is called TOPP-RA, time optimal path parametrization through reachability analysis [51]. It is a robotic motion planning library developed by Hung Pham of Nanyang Technological University. At a high level, the system works through discretization, subspace calculation and greedy selection. The path is discretized, and at each step, the reachable and controllable subspaces are computed. The highest velocity that lands the next state in a controllable subspace is selected, subject to the dynamic constraints entered by the user. The constraining factor is the desired maximum acceleration and velocity for each axis, or machine kinematic limits. A more rigorous description can be found in the original publication, but from a systems engineering standpoint, this component can be treated as a black box. TOPPRA takes in dynamic constraints and geometric paths and outputs discretized time optimal

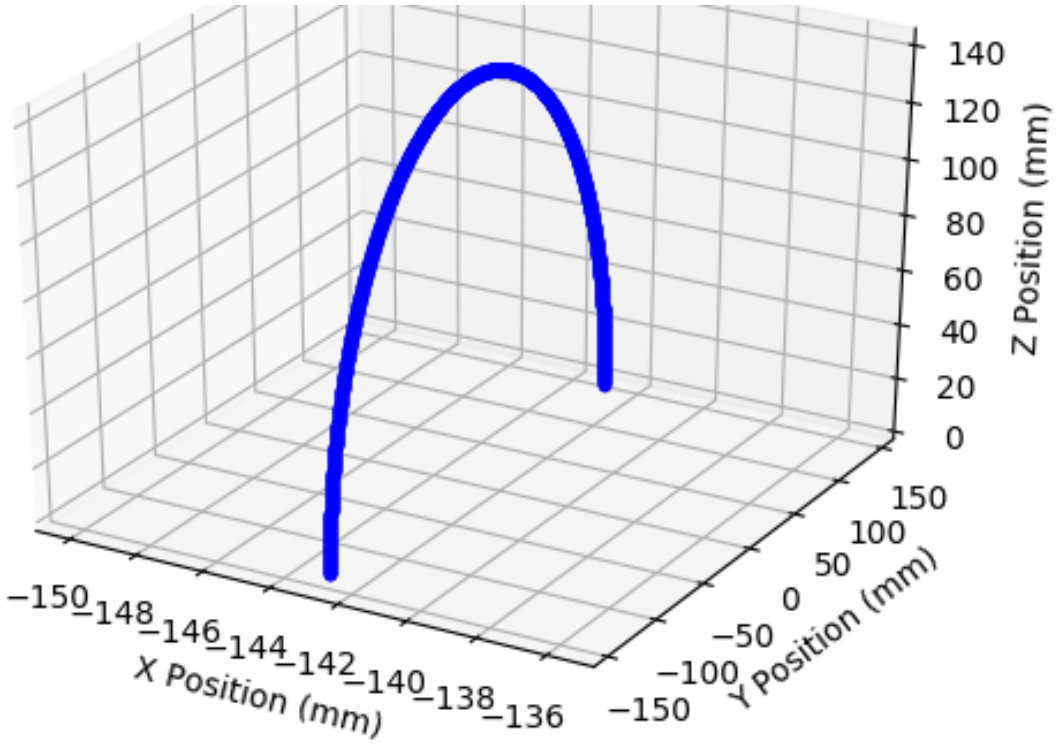

Figure 19. Original non-parametrized path in Cartesian space

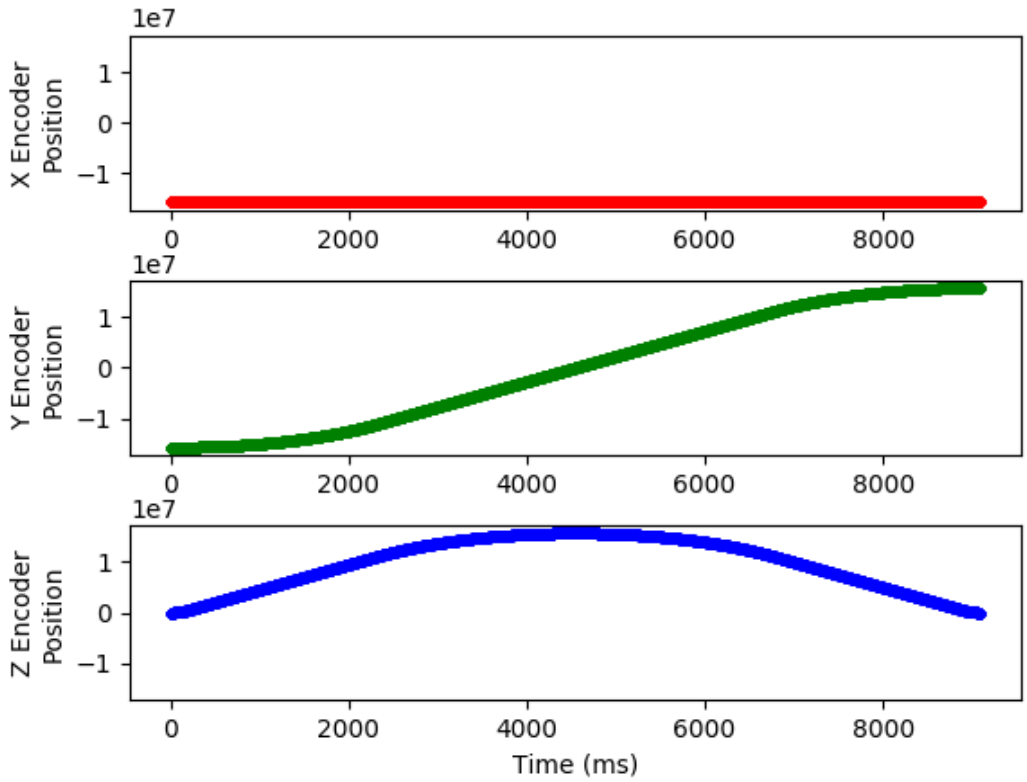

Figure 20. Time Parametrized paths for X, Y, and Z

parametrizations. As a simple example, Figure 19 shows a basic toolpath segment in Cartesian space, and Figure 20 shows the position set points for each axis after time optimal parametrization and resampling at 1 millisecond intervals. It is also worth noting that this module can work with higher degree of freedom toolpaths, such as the five degree of freedom paths that may be implemented in future work.

The last element of concern for the TOPPRA module is how to connect it to machine hardware. The interface code block runs a server socket to read in toolpaths from the internet, reparametrize them, and pipe them to the control loop running in kernel space. Where the toolpaths come from is described next. The Linux pipe, named pipe, or FIFO is an inter-process communication mechanism much like the loopback socket described in the previous section on the GUI. However, the loopback socket is bidirectional, while

information only flows one way through the pipe. In future work, the GUI will receive feedback from the machine, but the TOPPRA module will never need such feedback, hence the difference in deployment. Finally, there is of course a corresponding pipe read, parse and command block in the c code running in the kernel.

3.6.2 SculptPrint

SculptPrint is a powerful CAM software for generating toolpaths. For reference, some toolpaths generated in SculptPrint are shown in Figure 21. This lab has been working with the software for years. It began as a dissertation project by Dr. Tommy Tucker, but has been under continual development since his graduation. Additionally, in dissertation

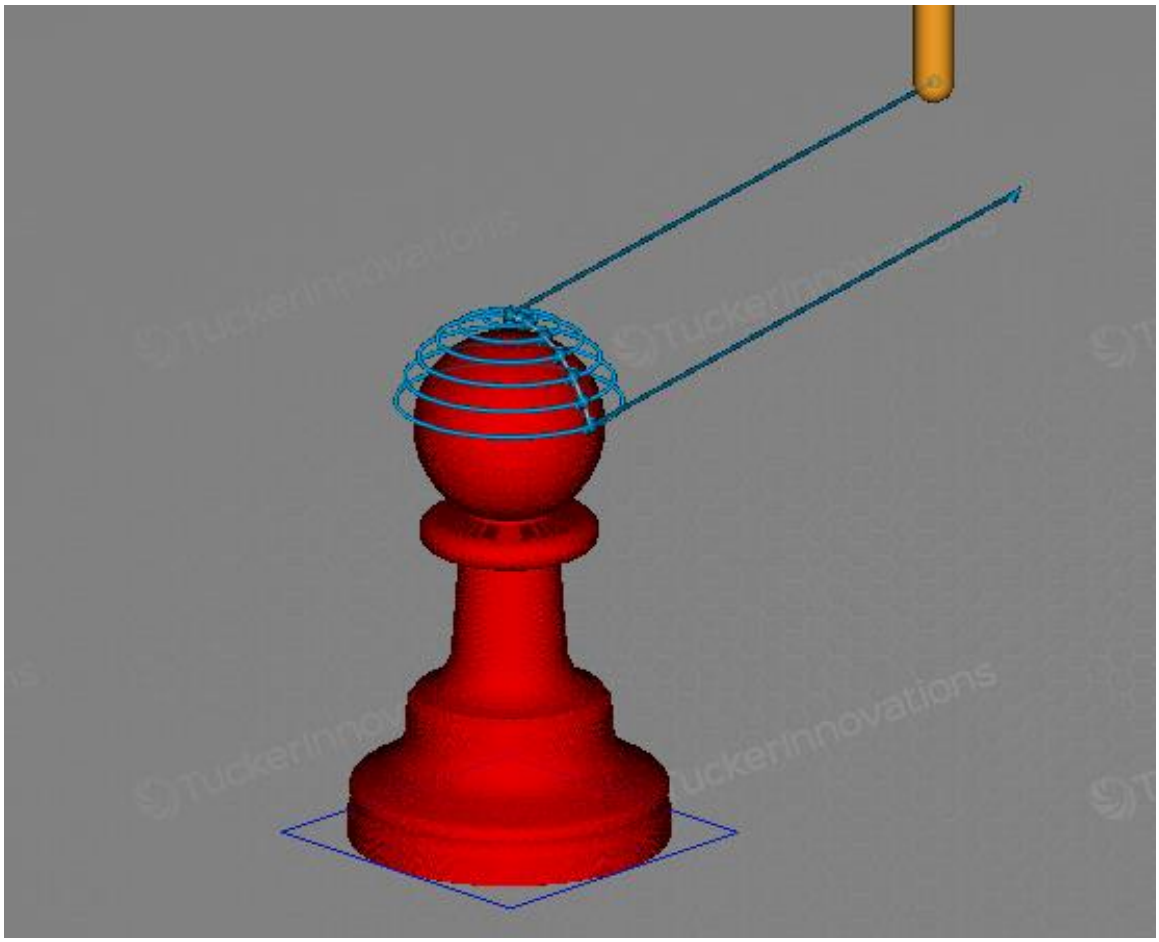

Figure 21. Example toolpaths generated in SculptPrint

work by Dr. Lynn, SculptPrint was used to control a PocketNC directly over the internet, while monitoring and simulating the process on an integrated digital twin [50]. The SculptPrint portion of this research is an extension on that body of work. The next few sections will introduce CAM software, highlight important features of SculptPrint, and proceed with new developments.

A CAM (Computer Aided Manufacturing) software is necessary to machine any non-trivial part geometry. The CAM package takes in stock and final part models and generates toolpaths, typically G Code, for execution on a CNC machine. Running these toolpaths removes successive layers of material until only the final geometry remains.

Many packages exist that are capable of toolpath generation, but none have the features of SculptPrint. The environment can simulate the machining process, integrate

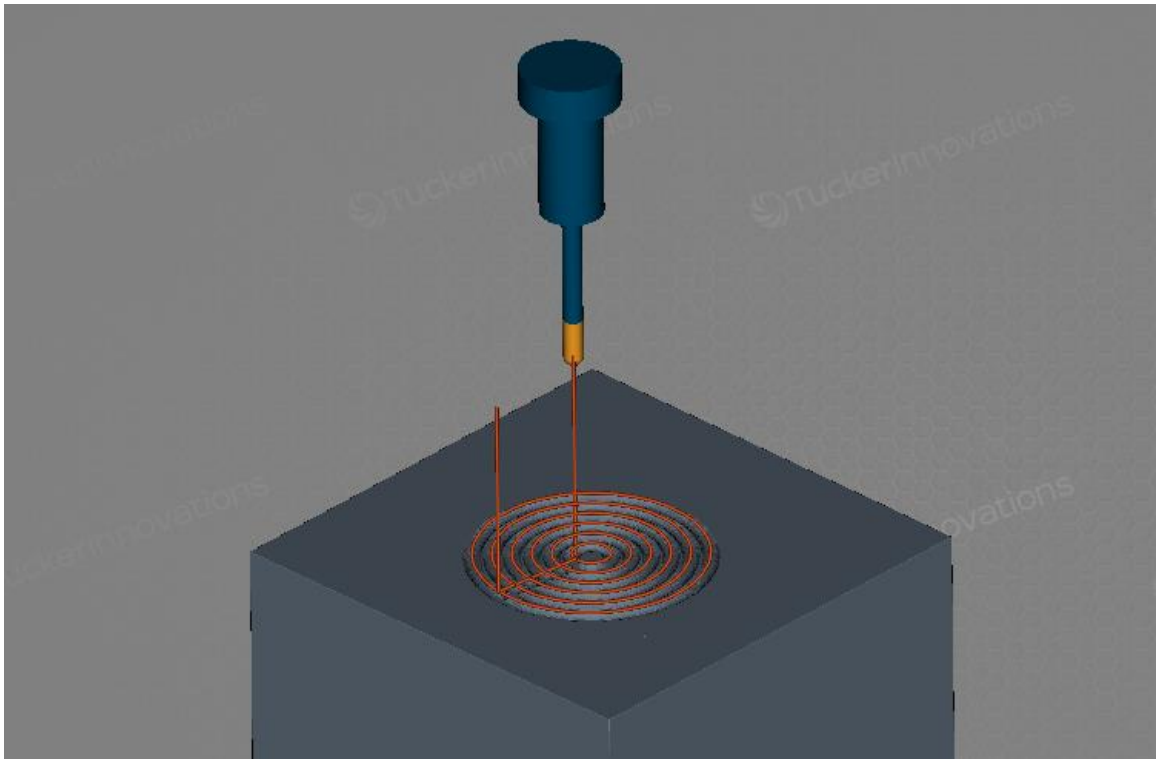

Figure 22. Simulated material removal in SculptPrint

new machine models, and incorporate general purpose python. The digital twin implemented on the research machine under consideration is based on these features.

The simulation capabilities of SculptPrint have been more thoroughly described in other publications, so they will only be presented briefly here. The software represents the part being machined using voxels, which are three dimensional discrete units analogous to pixels in digital photography. More classical CAM softwares use complex analytical functions in 3D space to represent part CAD models. The digital nature of the SculptPrint back end allows the software to simulate material removal on each tool pass. This functionality is shown for a simple toolpath in Figure 22. Analytical descriptions do not lend themselves to this level of fine grained simulation.

Another important simulation capability is interference checking. SculptPrint checks a given toolpath to ensure the tool doesn't collide with the part or the machine. This is less relevant for three axis machining, but is useful in higher degrees of freedom. As previously mentioned, a potential area of future work on this machine is the integration of a fourth and fifth axis turntable.

The digital twin model developed and integrated for this machine is shown in Figure 23. The process of integrating new models has been streamlined by the SculptPrint team. A Rhino plugin is used to import machine assembly CAD models. The plugin includes a kinematics engine that generates a set of homogeneous coordinate transforms. The transform chains range from ground to part fixture and ground to cutting tool. These transforms are used to generate the inverse kinematic equations that convert tool paths into actuator profiles. By the same method, forward kinematics are developed and used to simulate actuator profiles on the digital twin. For this three axis machine, the transforms

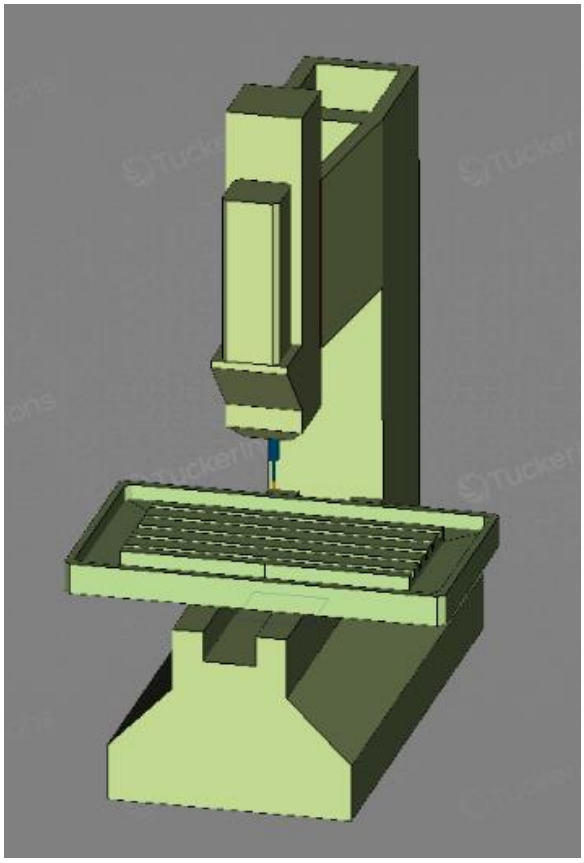

Figure 23. Model of the research machine integrated in SculptPrint

are relatively simple. However, the environment supports five axis machining as in the work by Dr. Lynn. As previously mentioned, a fourth and fifth axis may be integrated on this machine in the future.

The ability SculptPrint has to integrate general purpose python enables socket communication between the CAM computer and machine control computer over the internet. By this mechanism, the actuator profiles generated by inverse kinematics are used to control the connected machine remotely. Meanwhile, the control computer sends process data back to the digital twin, where the machining process is simulated. This is outlined in Figure 24. In this environment the planned and realized paths are traced out in 3D space for visual inspection. Additional feedback is also possible. For example, current consumed by the spindle motor can be sent back and displayed in the CAM environment as an indication of tool wear. The CAM environment also offers a number of buttons with callbacks in python. In future work, this interface could be used to more tightly integrate machine control with the CAM environment.

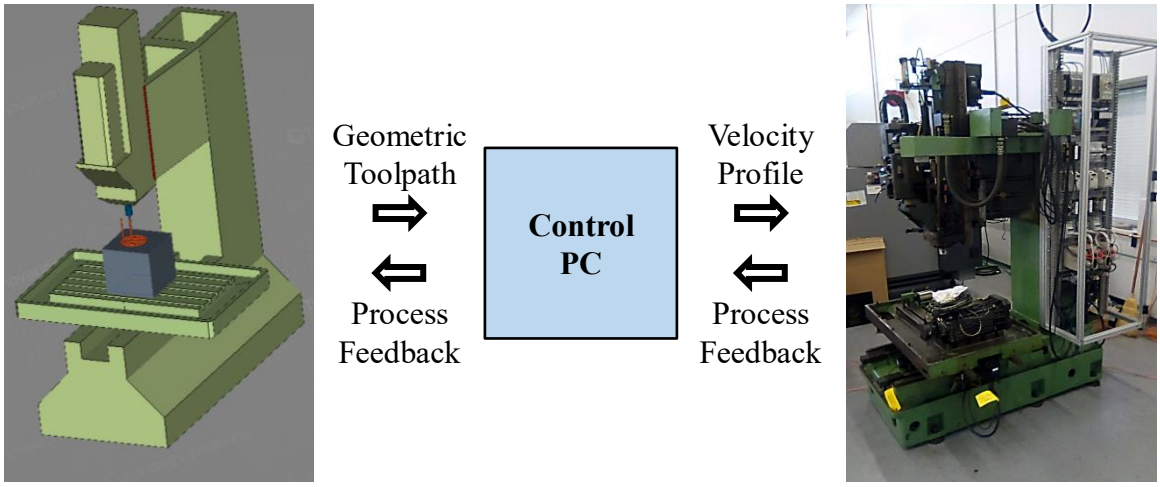

Figure 24. The digital twin integrated into SculptPrint and its relationship with hardware

CHAPTER 4 RESULTS AND DISCUSSION

The results for this project can be broken into engineering outcomes and research contributions. Both are presented in the following sections.

4.1 Functional Retrofit

The engineering outcomes can be summarized as the successful retrofit of an outdated CNC machine with a modern electrical system. This system includes the primary drive

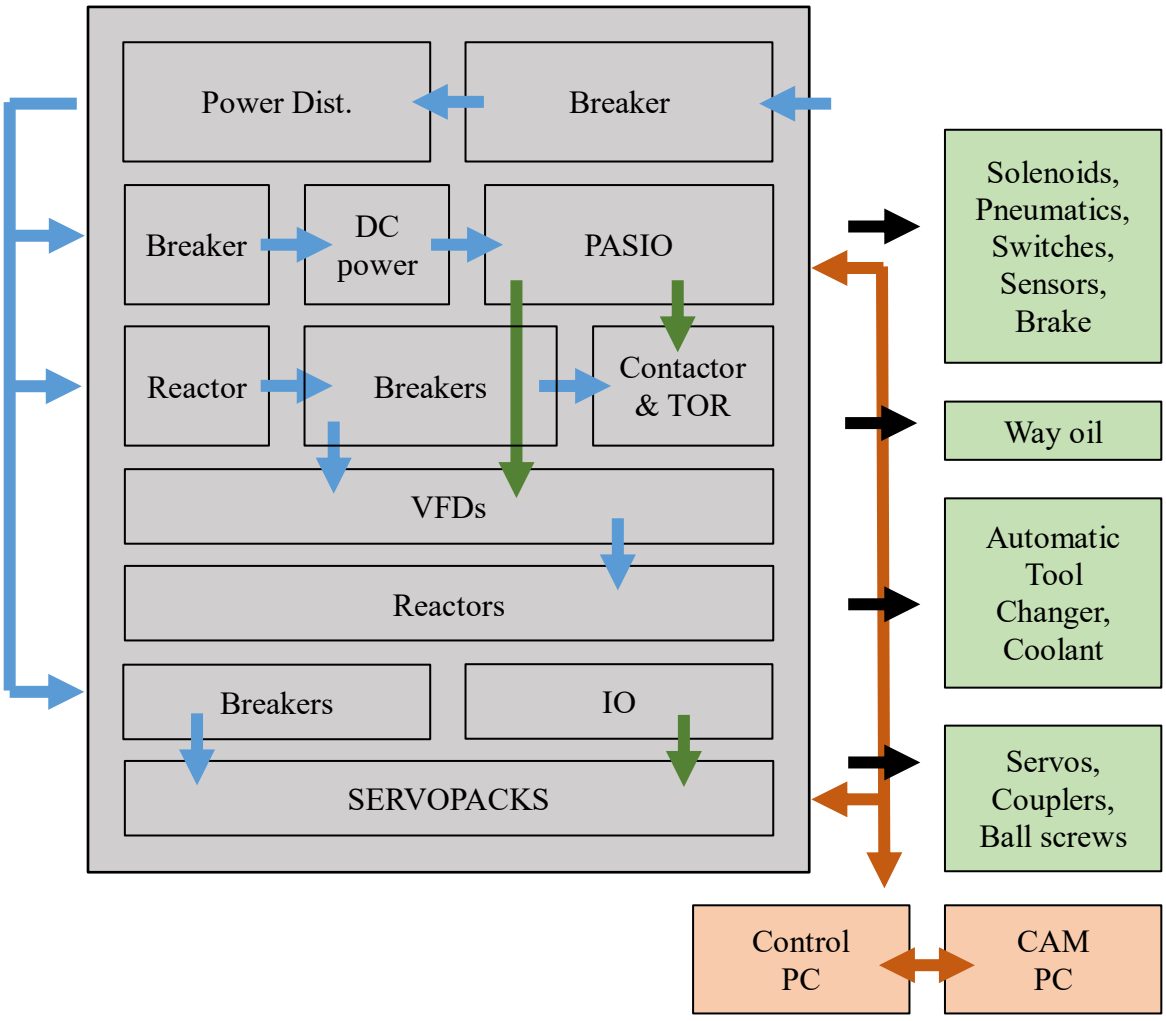

Figure 25. Summary of machine hardware: Blue arrows are power, green arrows are control signals, orange arrows are the communication network, and black arrows are machine outputs

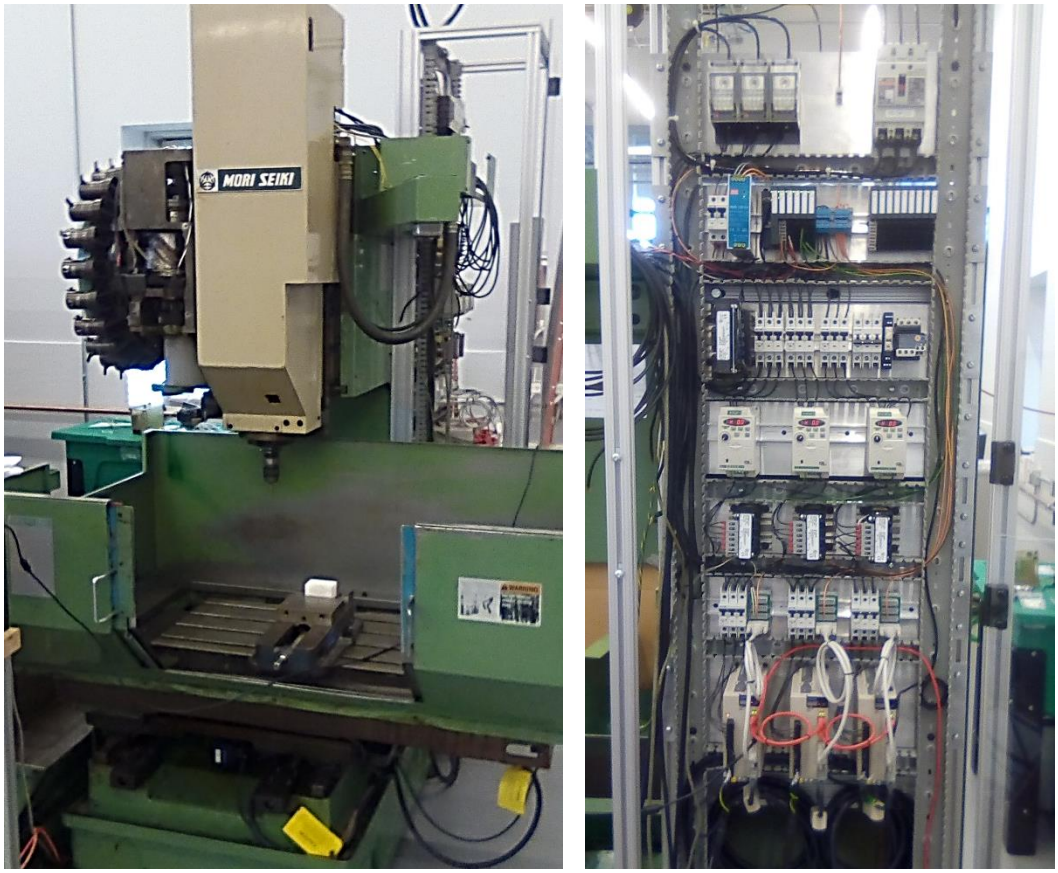

Figure 26. Present state of machine hardware (left) and control cabinet (right)

axes and a number of supporting subsystems. The design is summarized in Figure 25. The following sections go into further detail. Additionally, as research continues, the machine is beginning to come together again as a functional CNC. The present state of the machine is shown in Figure 26.

4.1.1 Functioning Subsystems

The system includes hardware for controlling coolant flow, switching way oil, and driving the automatic tool changer. The power distribution system includes three phase, single phase and DC components, along with high current analogue and digital IO, main and branch overcurrent and short circuit protection, as well as some current filtering where needed. The entire system is housed in an organized, modular 8020 frame. Complete

integration and software development for the way oil, coolant, and tool changer systems was demoted as less important to the research goals of the project, but the induction motors, VFDs and pneumatic system have all been tested and verified. The hardware is in place; this is an engineering outcome on its own. Further development is left to future work and to the research needs of the students who will inherit this machine.

4.1.2 Functional Drive System

The principal engineering outcome is the functionality of the primary drive system in three axes. The system can receive and interpret toolpaths, generate velocity profiles, and execute these profiles. The servos can communicate synchronously with one another and with the main controller, receiving new set points and returning process feedback every millisecond. In terms of a case study in CNC design, the mechanical hardware, electrical system, and supporting body of control and communication software is a successful outcome. One of the project goals was simply to retrofit an outdated CNC with modern servos, and this functionality is in place.

4.1.3 Drive System Performance

A ball-bar test was conducted to evaluate the performance of the drive system. A ball-bar is a rod shaped linear variable differential transformer or LVDT with spherical metal balls on each end. A magnetic insert is attached to the spindle, and a magnetic base is attached to the machining platform. The balls on either end of the rod attach to the insert and the base. The standard test is performed by executing circular paths in the orthogonal X, Y, and Z planes, while comparing the commanded position profiles to those physically achieved by machine hardware.

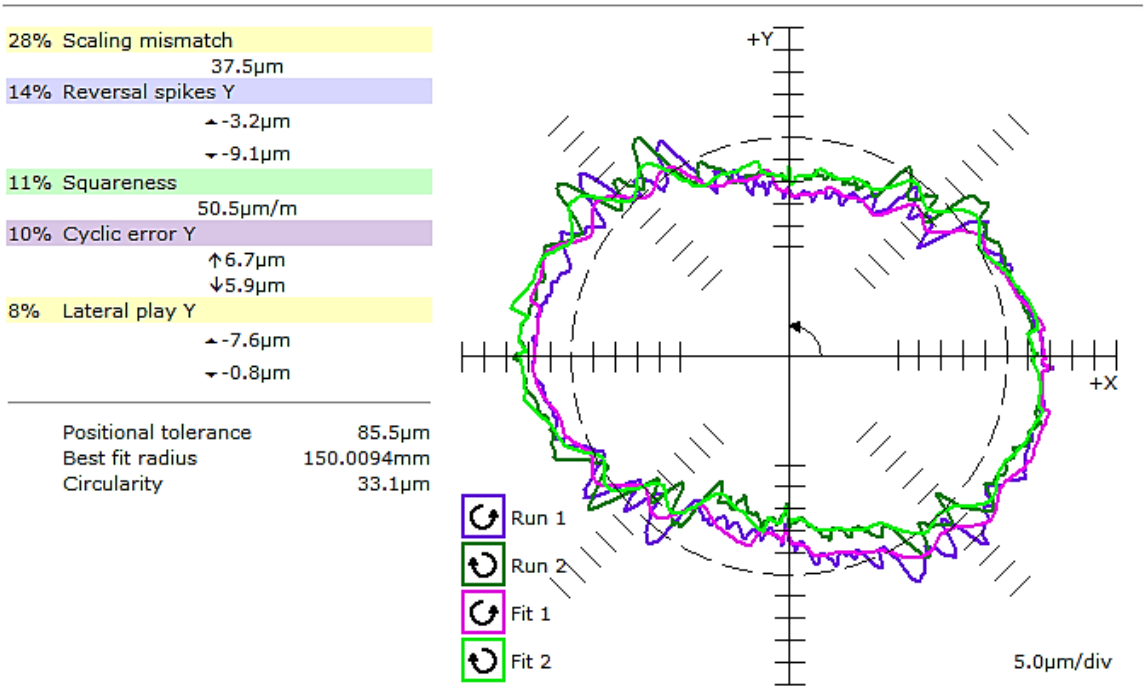

Figure 27. Ball bar test results

The results of this test, in the most informative horizontal plane, are displayed in Figure 27. The test showed a worst case positional tolerance of around 85 microns or 3 thousands of an inch for a circular path, with a 150 mm radius, in the horizontal plane, at 3000 mm/min. The modern state of the art for CNC machining is closer to 10 microns. However, the tolerance achieved by this first iteration is quite acceptable for many machining applications outside the realm of extremely high precision engineering.

The Renishaw ball-bar program generates the most likely causes for machining inaccuracy by algorithmically comparing the generated error profile with profile features from a collection of common sources. The three factors which contribute most, as identified by the Renishaw program, are scaling mismatch, Y axis reversal spikes, and squareness at 28%, 14%, and 11% respectively. Any of these mechanical defects seems plausible for a 30 year old machine. The cause of a scaling mismatch is a difference in travel along the

two axes. This can be caused by difference in pitch between ball screws, by differences in thermal expansion, by damage to one of the ball screws, or by linear rails that are not straight. None of these effects are large, but according to the test, the outcome of scaling mismatch is 37.5 µm of inaccuracy on a circular path with 150 mm radius. This affect could be compensated in software, after further testing to better characterize the phenomena. Reversal spikes indicate a momentary delay in motion when an axis changes direction of travel. The likely cause is friction in the linear guide ways. The force of friction changes directions at a reversal point. This introduces a step transition in the torque necessary for accurate tracking, so the axis temporarily stops. Completing the way oil integration may reduce this affect. Nonlinear feed forward torque compensation could also be integrated at transitions. Another mechanical error identified by the Renishaw program is cyclic error in the Y axis, said to be caused by a damaged ball screw or an error in mounting. This could also be caused by oscillatory behavior in the servo controller. In general, the ball bar test assumes a level of accuracy in the controller that may be unreasonable on this experimental platform, so further testing would be required to attribute error to mechanical sources with a high degree of confidence. Further suggestions for increasing accuracy are presented in the future work section.

4.1.4 Tracking Performance

In an effort to better characterize the performance of the current system, the ball bar test is performed at a range of speeds, while monitoring the error in the servo position control loop. Horizontal plane positional tolerance is recorded by the ball bar and compared with the worst case servo error over a full ball bar test in the X, Y, and Z planes. The horizontal plane was used for the ball bar results because this plane allows full 360º runs

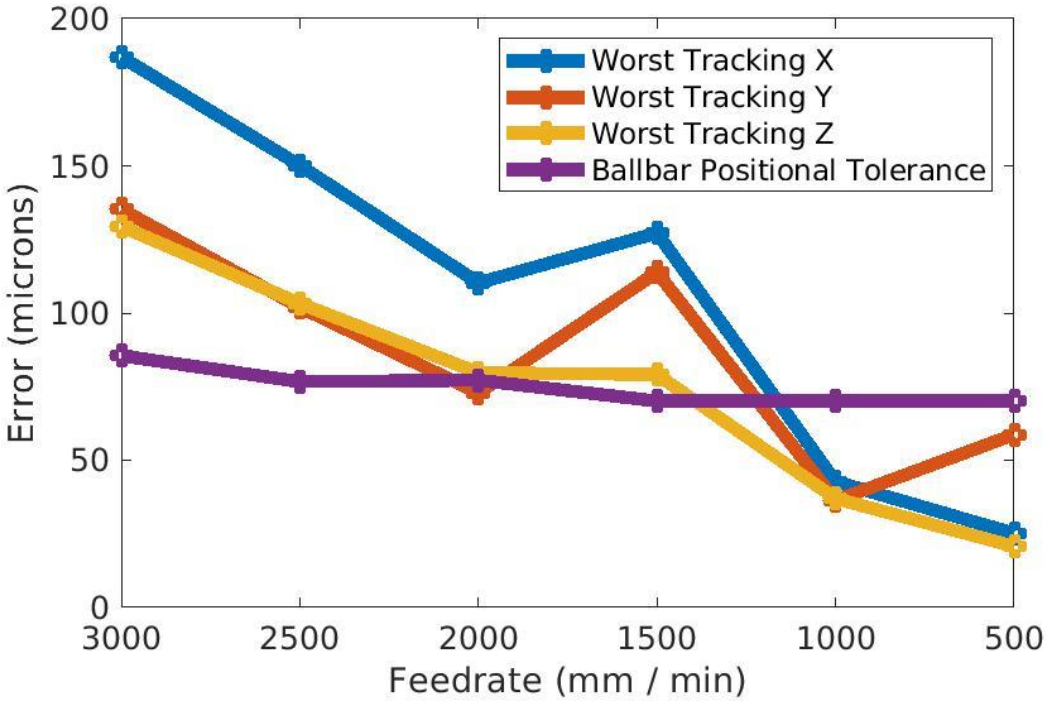

Figure 28. Comparison of ball bar and servo tracking error at a range of speeds

and is the only plane for which overall positional tolerance is reported. The results of this test are shown in Figure 28. The expectation was that servo error would make up some percentage of total error as reported by the ball bar. However, the data shows that servo error is greater than ball bar error at higher speeds. This is likely because the ball bar essentially measures the position reference tracking ability of the system, without consideration for the transient tracking performance. The servo error includes the information on how far the realized path lags behind the reference path in time. This lag depends on the rate of change of the position set point, as indicated by the slope of the servo error curves. The point of this project was not advanced controls research and implementation, so mitigation of servo tracking error is left to future work. This section is only meant to make clear the current limitations of the system. Some form of full state

feedback control that attempts to track velocity and position trajectories simultaneously would be a reasonable next step towards mitigation. An additional feedforward technique that offsets the trajectory set points in time by the quantifiable lag of the closed loop system would be another reasonable research direction.

4.2 Research Outcomes

The primary research contributions of the project are the generic hardware, open software controller, and the innovations this has enabled: internet based CAM integration, digital twin incorporation, assimilation of open source path planning software, and the elimination of G code. The software components at play are summarized in Figure 29. They are discussed further in subsequent sections.

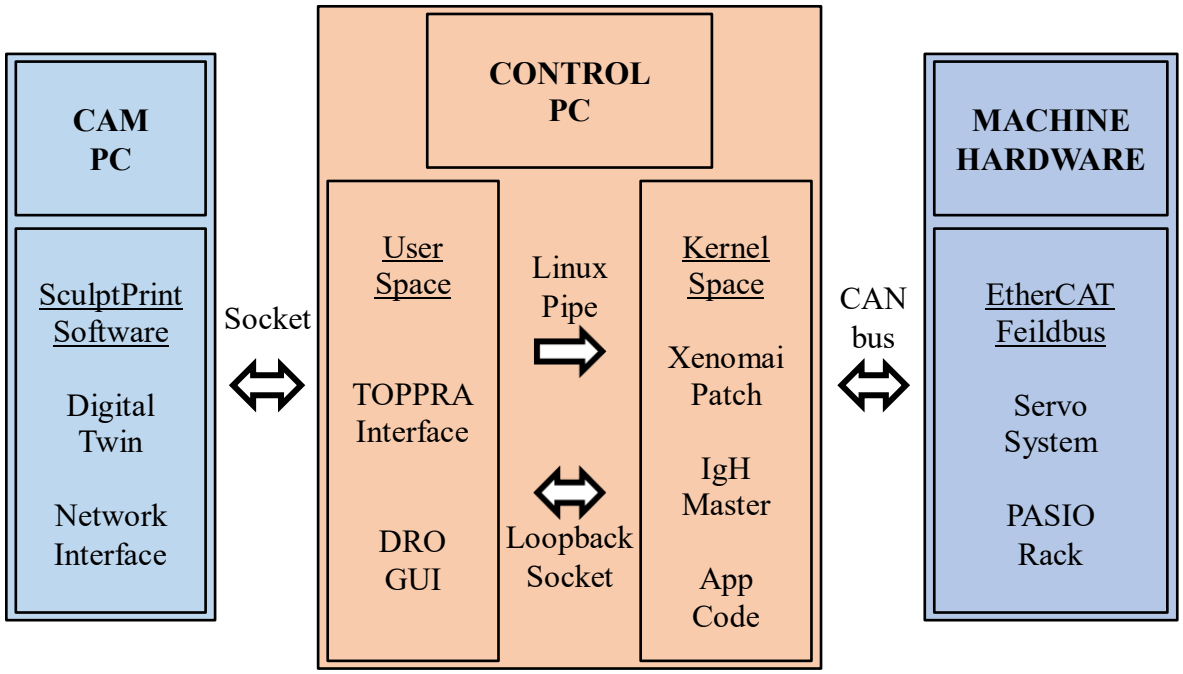

Figure 29. Control system block diagram

4.2.1 Open Source Controller

A standard PC running a patched Linux kernel can communicate and command all machine hardware programmatically. The system can assimilate unconstrained software in standard languages like python and C, and interface with the internet in an unimpeded manner.

4.2.2 Internet Integrated CAM

The open controller and the integration flexibility of SculptPrint enable a direct connection between the PC controlling machine hardware and the PC running the CAM environment via TCP sockets written in python. Toolpaths generated in the CAM environment are sent directly to machine hardware over the internet. This is a step towards the long standing research goal of integrating engineering and manufacturing in a unified, data rich environment. It also makes the dated, but still ubiquitous use of USB drives for transferring toolpath data unnecessary.

4.2.3 The Digital Twin

SculptPrint also enables the incorporation of a simulated version of the process and the machine in the CAM environment. Figure 30 shows the ball bar test being performed on hardware and simulated with encoder feedback on the digital twin.

A first order model of machine geometry and kinematics was developed and integrated into SculptPrint. The tool paths generated in CAM can be simulated on this model prior to execution. When ready for execution, the machine hardware can be commanded directly from this environment. The toolpaths are sent over the internet to the machine, while the machine sends back live process feedback. A digital version of the machining process takes place in the CAM environment, and the commanded and executed

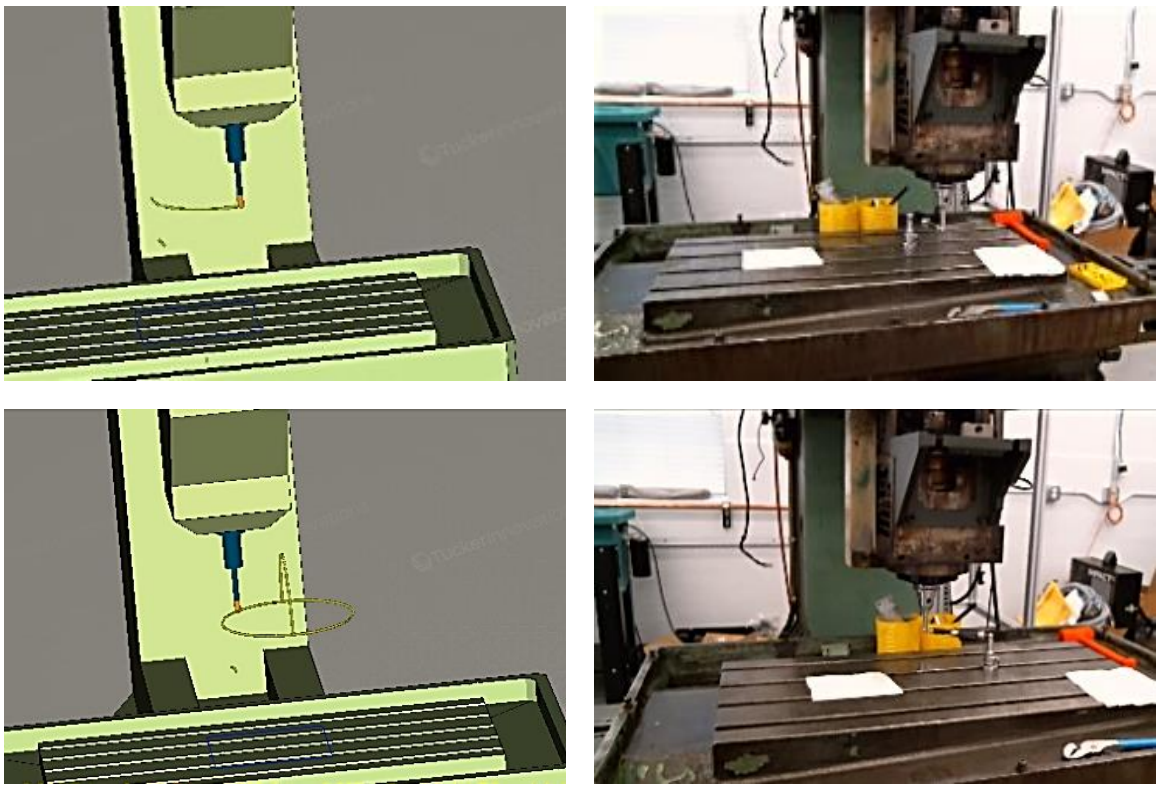

Figure 30. Two time stamps showing synchronous execution of the ball bar test on hardware and in simulation

toolpaths can be compared for quality assurance. Figure 31 shows an exaggerated example of this comparison. That data was taken during an aggressive tuning run, and the executed path is highly oscillatory. The data was also taken when the resolution of feedback was in millimeters; this has since been increased to encoder level precision.

This integration is only possible because the open machine controller can send, receive and process data arbitrarily, and because all machining data and controls are accessible within the software system.

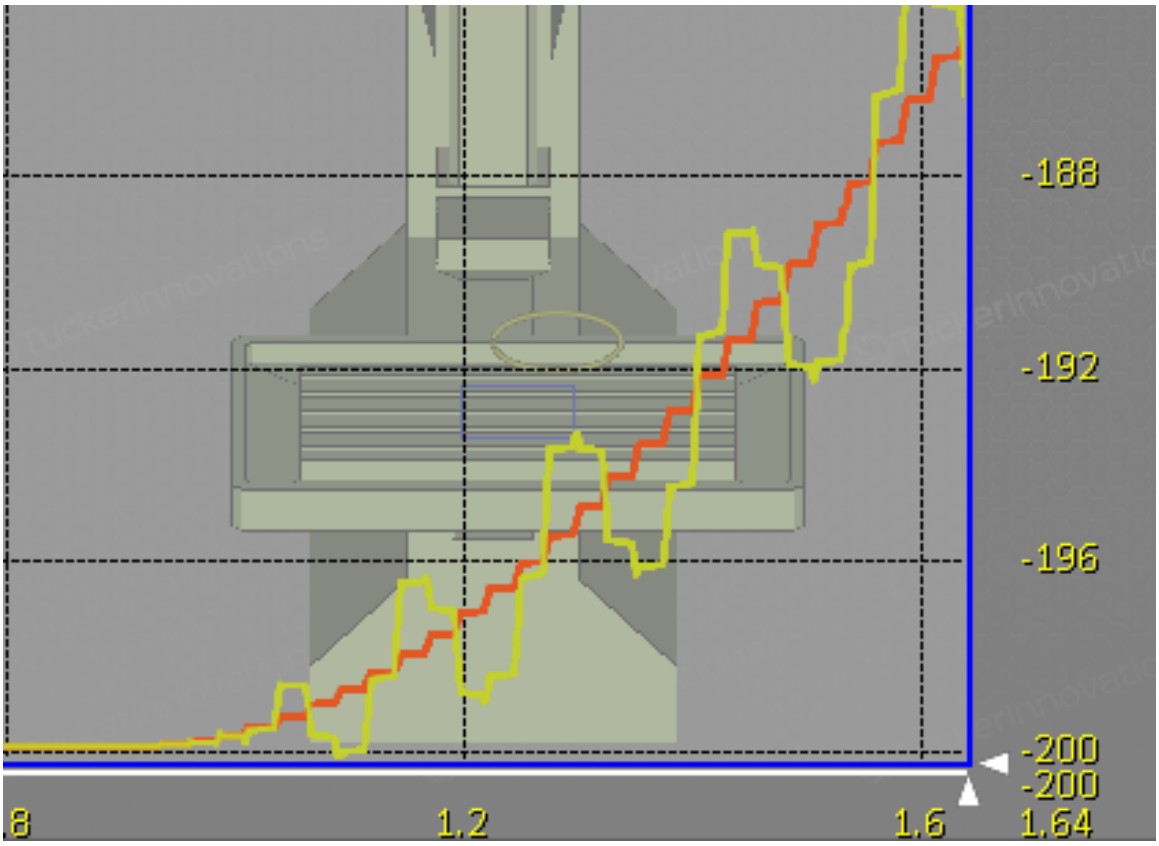

Figure 31. A comparison of planned (orange) and executed (yellow) toolpaths in millimeters vs seconds

4.2.4 Incorporated Open Source Software

When the machine controller receives the toolpaths from SculptPrint. They must be converted into velocity profiles before execution. This constrained optimization problem is in general non trivial. The open source nature of the controller and the ability to modify the functionality with standard programming languages enabled the incorporation of a body of python code written by other researchers. This is just one example of the research flexibility and potential for rapid improvement enabled by the platform. Additionally, the systems GUI was developed using an open source python package known as tkinter.

4.2.5 Absence of G Code

The G Code standard has been a bottleneck in CNC research for years, but it remains in place. This system presents an alternative in the form of more flexible and intuitive point clouds. The point clouds are fitted with splines in the controller, before time parametrization and execution. G Code typically contains a desired feed rate, but no acceleration information, and is executed in slightly different ways from machine to machine. The system employed here gives the user complete control over the trajectory. Additionally, G Code is typically run open loop, but this spline system can be updated in a closed loop way.

CHAPTER 5 CLOSING

This section will summarize the conclusions drawn from this body of research and outline directions for further and future investigation.

5.1 Retrofit and Industry 4.0

The first conclusion is that an 80's era CNC machine can be retrofitted with modern servos and communication systems, and therefore integrated into the industry 4.0 framework. With a relatively generic upgrade, the system achieved accuracy on the order of thousandths of an inch, but further development in the control software could likely reduce this error by a factor of two or more. This means that the ageing machines that make up a substantial component of US manufacturing can still be rendered valuable as the industry transitions towards industry 4.0.

5.2 Generic Controller

The second conclusion is that an arbitrary desktop running Linux can communicate with and control state of the art CNC systems. The PC hardware is a fraction of the cost of an industry standard controller. The open nature of the controller eliminates the need for the G Code interpreter and trajectory planner that make up a large portion of an industry controller. The desktop instead uses a more flexible and intuitive point cloud system, with a free trajectory planner developed by another research group. The desktop uses a network interface and the CAN Open over EtherCAT communication standard rather than proprietary communication protocols. The desktop enables the development of user interfaces in software, thus eliminating the need for another expensive feature of the industry controller. An industry controller is essentially a means of interpreting G Code, generating trajectories, controlling machine hardware, and providing a user interface.

While the desktop version is of course far less developed and robust when compared to its industry counterpart, it can still accomplish all these functions. More importantly, it does so cheaply and in an open manner that facilitates that data and software integration and communication goals of Industry 4.0.

5.3 Digital Twin and CAM Integration

An additional conclusion is that the SculptPrint environment provides a feasible platform for digital twin integration of the CNC machine and process. The platform has the flexibility to incorporate new models. The system also provides the software flexibility for remote control and monitoring of an arbitrary CNC system, provided the CNC system is equally open, over the internet by means of standard python. Additionally, the point cloud system employed by this CAM environment is evidently a viable alternative to standard G Code.

5.4 Future work

The next few sections present a few near and far term directions for potentially fruitful research along the lines of the present project.

5.4.1 Controller Improvements

Further work on tuning the servos and introducing additional feedforward techniques, as discussed in the sections on servo control and tracking performance, could potentially increase accuracy. Furthermore, the system update time of one millisecond could likely be decreased. The EtherCAT network claims one microsecond update times. The code in the main real time loop would have to be optimized for speed or outsourced to separate threads where possible, but the main loop time could likely be lowered. This should be explored as a means of increasing accuracy, as higher bandwidth control

generally performs better. More broadly, focused research and development on state of the art CNC control algorithms could lead to improved performance.

5.4.2 Spindle Motor Integration

An additional short term goal is the integration of the supporting power systems for the spindle motor. This body of work focused on the control of the machine, rather than actual manufacturing, but integration of the spindle motor is an essential next step. Again, most of this system has already been designed. Integration in hardware and software development are the next steps.

5.4.3 Hybrid Manufacturing

The immediate plan for continued research is the integration of a MIG welder. This will leverage the open nature of the platform in an investigation of low cost, flexible, hybrid manufacturing. The SculptPrint environment provides support for additive as well as subtractive manufacturing, so this link will continue to be explored. The flexibility of the system will make closed loop feedback and dynamic replanning feasible.

5.4.4 Five Axis Capability

Another near term goal is the integration of a fourth and fifth axis turn table to expand the capabilities of the system. The turn table is shown in Figure 32. The majority of the supporting power systems have been spec'd already.

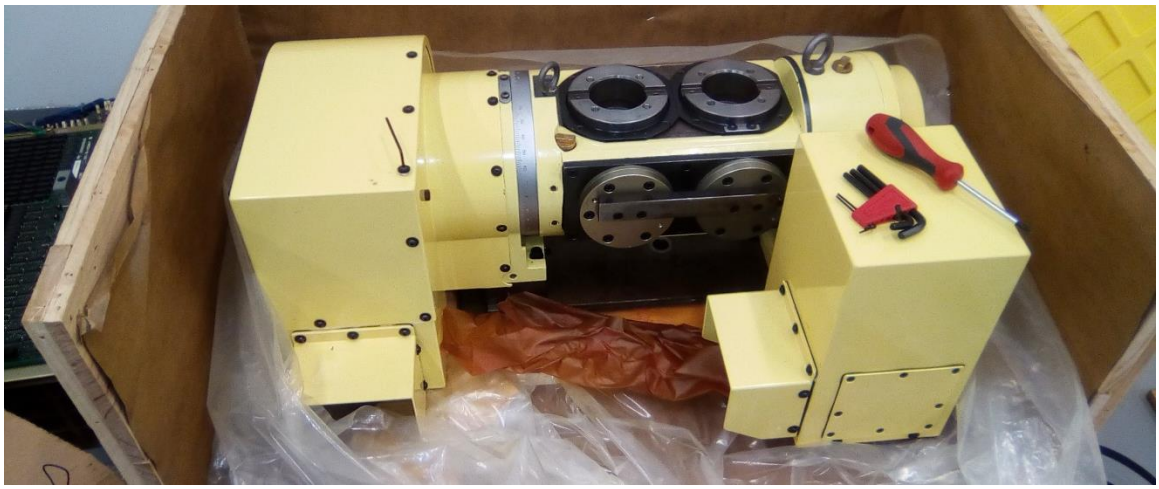

Figure 32. The turn table intended to provide five axis capabilities

5.4.5 Single Board Controller

The requirements on the PC controlling the machine are quite low: modest RAM, modest storage space, an Ethernet port, a USB port, the ability to run Linux. As previously mentioned, part of the development for this system was performed on a Beaglebone, shown in Figure 33. This was dropped in favor of the increased usability and memory of a desktop.

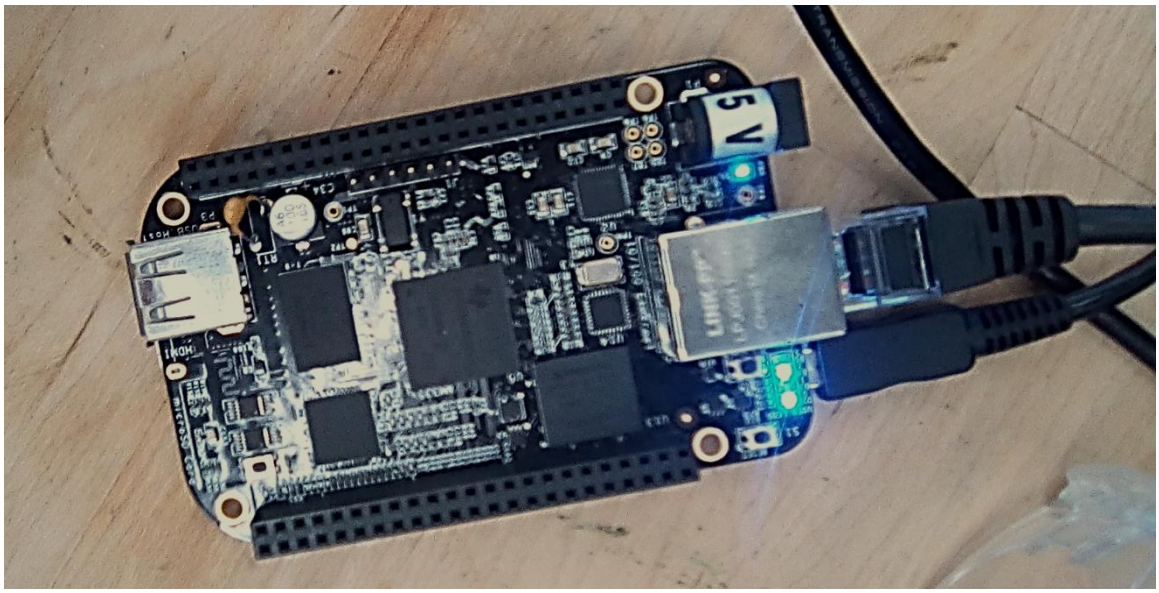

Figure 33. The Beaglebone used for part of development

However, the final controller could likely be implemented, at even lower cost, on a Beaglebone, Raspberry Pi, or other single board computer. This is a potentially interesting line of future research.

5.4.6 Robustness, Generality and Benchmarking

A more ambiguous, long term goal is increased robustness and generality along with better benchmarking of the controllers performance. An industry standard controller, along with other open source alternatives such as EMC, has the obvious advantage in terms of robustness and benchmarking. If this controller is actually to gain traction, the code and the machine should be subjected to rigorous testing. Generality in the form of a software base applicable to a larger array of computer hardware and CNC systems would also be desirable. However, this robustness and generality come at the price of a larger, less flexible code base. The present system is optimized for the research needs of the lab. As the system continues to develop, benchmarking things like manufacturing quality and machining time against standard machines running G Code will become increasingly relevant.

5.4.7 App Store for Research

An additional long term goal would be the integration of an app store like environment where software developed by manufacturing research groups could be easily integrated into the open source controller. This is already possible in the form of python modules, like the ones currently in use for trajectory planning and the GUI. However, a standardized system of some kind could make the integration of software on arbitrary machines easier. For example, it is conceivable to imagine anti-spindle chatter software, or closed loop feed optimization software being integrated in a straight forward way.

CITATIONS

1. Thoben, K.-D., S. Wiesner, and T. Wuest, *"Industrie 4.0" and smart manufacturing-a review of research issues and application examples.* International Journal of Automation Technology, 2017. **11**(1): p. 4-16.
2. Kang, H.S., et al., *Smart manufacturing: Past research, present findings, and future directions.* International Journal of Precision Engineering and Manufacturing-Green Technology, 2016. **3**(1): p. 111-128.
3. Lasi, H., et al., *Industry 4.0.* Business & information systems engineering, 2014. **6**(4): p. 239-242.
4. Khaitan, S.K. and J.D. McCalley, *Design techniques and applications of cyberphysical systems: A survey.* IEEE Systems Journal, 2014. **9**(2): p. 350-365.
5. Esmaeilian, B., S. Behdad, and B. Wang, *The evolution and future of manufacturing: A review.* Journal of Manufacturing Systems, 2016. **39**: p. 79-100.
6. Scholten, B., *The road to integration: A guide to applying the ISA-95 standard in manufacturing*. 2007: Isa.
7. Leitner, S.-H. and W. Mahnke, *OPC UA–service-oriented architecture for industrial applications.* ABB Corporate Research Center, 2006. **48**: p. 61-66.
8. Vijayaraghavan, A., et al., *Improving machine tool interoperability using standardized interface protocols: MT connect.* 2008.
9. Pezzullo, V., *Design of a custom software application to monitor and communicate cnc machining process information to aid in chatter identification.* 2014.

10. Proctor, F.M. and J. Michaloski, *Enhanced machine controller architecture overview*. 1993: US Department of Commerce, National Institute of Standards and Technology.

11. Proctor, F.M., et al., *Open architectures for machine control.* NIST Report, 1993.

12. Schofield, S. and P. Wright, *Open architecture controllers for machine tools, Part 1: Design principles.* Journal of Manufacturing Science and Engineering, 1998. **120**(2): p. 417-424.

13. Koren, Y., et al., *Reconfigurable manufacturing systems.* CIRP annals, 1999. **48**(2): p. 527-540.

14. Correa, J.E., N. Toombs, and P.M. Ferreira, *A modular-architecture controller for CNC systems based on open-source electronics.* Journal of Manufacturing Systems, 2017. **44**: p. 317-323.

15. Association, E.I., *EIA Standard EIA-274-D interchangeable variable block data format for positioning, contouring, and contouring/positioning numerically controlled machines*. 1979, Electronic Industries Association Arlington, VA.

16. Xu, X.W. and S.T. Newman, *Making CNC machine tools more open, interoperable and intelligent—a review of the technologies.* Computers in Industry, 2006. **57**(2): p. 141-152.

17. Hardwick, M., *On STEP-NC and the complexities of product data integration.* Journal of Computing and Information Science in Engineering, 2004. **4**(1): p. 60-67.

18. Jerard, R.B. and O. Ryou, *NCML: a data exchange format for internet-based machining.* International journal of computer applications in technology, 2006. **26**(1): p. 75-82.

19. Zhang, Q., S. Li, and J. Guo, *Smooth time-optimal tool trajectory generation for CNC manufacturing systems.* Journal of manufacturing Systems, 2012. **31**(3): p. 280-287.

20. Timar, S.D., et al., *Algorithms for time–optimal control of CNC machines along curved tool paths.* Robotics and Computer-Integrated Manufacturing, 2005. **21**(1): p. 37-53.

21. Dong, J., P. Ferreira, and J.A. Stori, *Feed-rate optimization with jerk constraints for generating minimum-time trajectories.* International Journal of Machine Tools and Manufacture, 2007. **47**(12-13): p. 1941-1955.

22. Erkorkmaz, K. and M. Heng, *A heuristic feedrate optimization strategy for NURBS toolpaths.* CIRP annals, 2008. **57**(1): p. 407-410.

23. Wang, F.-C. and P. Wright, *Open architecture controllers for machine tools, part 2: a real time quintic spline interpolator.* Journal of manufacturing science and engineering, 1998. **120**(2): p. 425-432.

24. Roach, R.A., et al., *Born Qualified Grand Challenge LDRD Final Report*. 2018, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Sandia ….

25. Soucy, K.A., *In-process monitoring for quality assurance of automated composite fabrication*, in *Review of progress in quantitative nondestructive evaluation*. 1996, Springer. p. 2225-2231.

26. Munasinghe, W., *Modular CNC design for intelligent machining, Part 2: Modular integration of sensor based milling process monitoring and control tasks.* Journal of Manufacturing Science and Engineering, 1996. **118**(4): p. 514-521.

27. Soori, M., B. Arezoo, and M. Habibi, *Tool deflection error of three-axis computer numerical control milling machines, monitoring and minimizing by a virtual machining system.* Journal of Manufacturing Science and Engineering, 2016. **138**(8): p. 081005.

28. Li, J., et al., *A quality prediction framework for multistage machining processes driven by an engineering model and variation propagation model.* Journal of Manufacturing Science and Engineering, 2007. **129**(6): p. 1088-1100.

29. Wang, L., W. Shen, and S. Lang, *Wise-ShopFloor: A web-based and sensor-driven e-shop floor.* Journal of Computing and Information Science in Engineering, 2004. **4**(1): p. 56-60.

30. Glaessgen, E. and D. Stargel. *The digital twin paradigm for future NASA and US Air Force vehicles*. in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*. 2012.

31. Kritzinger, W., et al., *Digital Twin in manufacturing: A categorical literature review and classification.* IFAC-PapersOnLine, 2018. **51**(11): p. 1016-1022.

32. Kannan, K. and N. Arunachalam, *A Digital Twin for Grinding Wheel: An Information Sharing Platform for Sustainable Grinding Process.* Journal of Manufacturing Science and Engineering, 2019. **141**(2): p. 021015.

33. Söderberg, R., et al., *Toward a Digital Twin for real-time geometry assurance in individualized production.* CIRP Annals, 2017. **66**(1): p. 137-140.

34. Brundage, M.P., et al., *Where Do We Start? Guidance for Technology Implementation in Maintenance Management for Manufacturing.* Journal of Manufacturing Science and Engineering, 2019. **141**(9).

35. Rosen, R., et al., *About the importance of autonomy and digital twins for the future of manufacturing.* IFAC-PapersOnLine, 2015. **48**(3): p. 567-572.

36. Boschert, S. and R. Rosen, *Digital twin—the simulation aspect*, in *Mechatronic Futures*. 2016, Springer. p. 59-74.

37. Xu, X., *Machine Tool 4.0 for the new era of manufacturing.* The International Journal of Advanced Manufacturing Technology, 2017. **92**(5-8): p. 1893-1900.

38. *SGD7S EtherCAT*. Available from: https://www.yaskawa.com/products/motion/sigma-7-servo-products/servopacks/sgd7s-ethercat.

39. *SGM7G*.

40. Johnson, C., *An introduction to flexible couplings.* World Pumps, 1996. **1996**(363): p. 38-43.

41. *Circuit Breakers, Fuses, and Disconnect Switches*. Available from: https://www.automationdirect.com/adc/overview/catalog/circuit_protection_-z-_fuses_-z-_disconnects.

42. *AC Line Reactors*. Available from: https://www.automationdirect.com/adc/overview/catalog/drives_-a-_soft_starters/ac_variable_frequency_drives_(vfd)/high-

performance/ac_line_reactors?gclid=CjwKCAjw_uDsBRAMEiwAaFiHa_3Jw-G5M3_EJb-95ZtrbhgtGPd-I7O3d8S-eToZ_0sgAck-8kZD4hoCUyQQAvD_BwE#bodycontentppc.

43. *I/O MODULES PASIO.* Available from: https://www.powerautomation.de/fileadmin/downloads/Brochures/EN/ITE_PA_PASIO_WEB_EN_20190708.pdf.

44. Chen, Y., et al. *The relevant research of CoE protocol in EtherCAT Industrial Ethernet*. in *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*. 2010. IEEE.

45. *EtherCAT - the Ethernet Fieldbus.*

46. *CANopen – The standardized embedded network.*

47. *Xenomai Realtime System.*; Available from: https://xenomai.org/.

48. Pose, D.-I.F.F., *IgH Master 1.5. 0 Documentation.* Ingenieurgemeinschaft IgH, 2013.

49. *Tkinter*. Available from: https://docs.python.org/3/library/tkinter.html.

50. Lynn, R., *Direct Servo Control of Positional Derivatives for 5-Axis CNC Machine Tools Using Densely-Sampled Toolpaths*. 2019, Georgia Institute of Technology.

51. Pham, H. and Q.-C. Pham, *A new approach to time-optimal path parameterization based on reachability analysis.* IEEE Transactions on Robotics, 2018. **34**(3): p. 645-659.